A Computational Theory of Grounding in Natural Language Conversation

David R. Traum

Technical Report 545 and Ph.D. Thesis
December 1994

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

A Computational Theory of Grounding in Natural Language Conversation

by

David R. Traum

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor James F. Allen

Department of Computer Science
College of Arts and Science

University of Rochester
Rochester, New York

1994

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	technical report

4. TITLE AND SUBTITLE

A Computational Theory of Grounding in Natural Language Conversation

5. FUNDING NUMBERS

N00014-92-J-1512, N00014-90-J-1811

6. AUTHOR(S)

David R. Traum

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES

Computer Science Dept.
734 Computer Studies Bldg.
University of Rochester
Rochester NY 14627-0226

8. PERFORMING ORGANIZATION

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESSES(ES)

Office of Naval Research
Information Systems
Arlington VA 22217

ARPA
3701 N. Fairfax Drive
Arlington VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

TR 545

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution of this document is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

(see title page)

14. SUBJECT TERMS

natural language; grounding; computational theory; speech act theory; situation-theoretic model

15. NUMBER OF PAGES

195 pages

16. PRICE CODE

free to sponsors; else $8.00

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	UL

Curriculum Vitae

David Rood Traum [redacted] Shortly thereafter, he began to use and understand natural language, little realizing that decades later he would be spending so much of his time analyzing how it works. He moved with his family to South Florida in 1968, and remained until leaving for Cambridge, Massachusetts to attend Harvard College in 1981. He graduated with a Bachelor of Arts in Applied Mathematics in 1985.

After short stays in Europe, back in Cambridge, and Los Angeles, he decided to pursue further study and went to San Jose State University, receiving a Master of Science in Computer Science – Mathematics in December, 1987. While there, he developed an interest in research while working on problems in graph theory with Sin-Min Lee, and an interest in Computational Linguistics from a course and discussions with Sam Mchombo. These interests were then fused after attending the LSA Linguistic Institute at Stanford University in the summer of 1987.

He enrolled in the PhD program at the University of Rochester in 1988, receiving along the way, through no fault of his own, a second Master's degree in 1990. While at Rochester, he was a teaching assistant for courses in data structures and AI programming, and a research assistant for James Allen. He now hopes that this document will allow him to complete yet another degree and move on once again.

Acknowledgments

After spending so much time in graduate school, trying to thank all of the people who have helped along the way seems almost as daunting a task as writing the dissertation itself. First, and foremost, I'd like to thank my advisor James Allen. He first introduced to me action-based approaches towards language and has been a constant collaborator in all of my work since. He has often helped me find the way through the difficult times, and his continual push for clarity and simplicity has allowed me to get much further than it often seemed I could.

I also want to thank the other members of my thesis committee. Len Schubert's clear insight and perspective were of great help in assessing the merits of both my own work and that of others. He also taught me a lot about semantic approaches to language and why we should care about them, even though the pragmatics is what we're "really" after. Mike Tanenhaus provided perspective from another branch of cognitive science, helping me understand some of the psycholinguistic approaches to discourse processing. Graeme Hirst shared an interest in trying to apply the insights from Conversation Analysis within a computational perspective and was very helpful in suggesting relevant work I hadn't yet come across. Graeme has also taken pains from soon after meeting me to introduce me to "kindred spirits" in the research community (including some of his own students), from which I've received great benefit.

My collaborators also deserve much credit. Alice Kyburg, Elizabeth Hinkelman, Keri Jackson, Lou Hoebel, Stephane Guez, and James were a great group to work with on a language system before TRAINS. Nat Martin and Stephen Feist built the Armtrak simulator that allowed me to explore ideas about action in a multi-agent environment in a concrete fashion. Frustrations in trying to use existing speech act taxonomies to label the dialogues Shin'ya Nakajima had collected and was studying the prosodic features of led to the formulation of the conversation act theory presented here. Derek Gross and Massimo Poesio also made valuable contributions to the early development. Elizabeth suggested that this would be a good framework within which to frame arguments about non-literal speech acts. Text we subsequently passed back and forth between Rochester and first Chicago and later Saarbrücken ended up here as Chapter 4. Working on the TRAINS system has been extremely valuable along a number of dimensions. James and Len deserve a large amount of credit for getting so many strong-willed, independent, and creative people to work so well together so productively. Interacting so closely with George Ferguson, Peter Heeman, and Massimo helped refine lots of ideas on the specifics of interaction between the implementation described in Chapter 6 and adjacent parts of the system. Chung Hee Hwang, Graham Katz, Janet Hitzeman, Marc Light, Nat,

and Tsuneaki Kato also made important contributions to the functioning of the whole system.

I would also like to thank many colleagues for stimulating conversations and comments on previous expositions of my work. It is impossible to note everyone who deserves mention here, but Susan Brennan, Phil Cohen, Julia Hirschberg, Ed Hovy, Karen Lochbaum, Susan McRoy, and Candy Sidner have been particularly helpful to me in sharpening some of the ideas presented here.

The Computer Science department at Rochester has been a great place to work, which I shall surely miss. The faculty and students work together in a supportive, and productive manner, but are always ready to go off and engage in needed diversion. All of the staff has been responsive and helpful. Many thanks to Peggy Frantz, Jill Forster, Peggy Meeker, Pat Marshall, Luidy Bukys, Jim Roche, and Tim Becker. Special thanks to Brad Miller who responded extremely promptly to sometimes inadequate bug reports and, in order to accommodate my needs, made many extensions to RHET beyond what it was ever intended to do.

The Cognitive Science group at Rochester has been equally beneficial, and I greatly value the interactions I've had with this group. Tom Bever deserves special commendations for inviting great visiting speakers over to his house for "students only" dinners. This allowed not only getting to know and talk to prominent scientists in a more informal setting, but built a great community among us students. I also want to mention Joel Lachter as an important influence from the beginning who was always willing to talk about the larger issues, as well as run a great bridge game.

This material is based upon work supported by ONR/DARPA under grant number N00014-92-J-1512, by ONR under research grant number N00014-90-J-1811, and by NSF under grant number IRI-9003841.

Finally, I'd like to thank my family for not forgetting about me during the busy times, and for accepting the "wrong kind" of doctor.

Abstract

The process of adding to the common ground between conversational participants (called *grounding*) has previously been either oversimplified or studied in an off-line manner. This dissertation presents a *computational* theory, in which a protocol is presented which can be used to determine, for any given state of the conversation, whether material has been grounded or what it would take to ground the material. This protocol is related to the mental states of participating agents, showing the motivations for performing particular grounding acts and what their effects will be.

We extend speech act theory to account for levels of action both above and below the sentence level, including the level of *grounding acts* described above. Traditional illocutionary acts are now seen to be multi-agent acts which must be grounded to have their usual effects.

A conversational agent model is provided, showing how grounding fits in naturally with the other functions that an agent must perform in engaging in conversation. These ideas are implemented within the TRAINS conversation system.

Also presented is a situation-theoretic model of plan execution relations, giving definitions of what it means for an action to begin, continue, complete, or repair the execution of a plan. This framework is then used to provide precise definitions of the grounding acts in terms of agents executing a general *communication plan* in which one agent must present the content and another acknowledge it.

Table of Contents

List of Tables

List of Figures

1 Introduction

Clark [1992] discusses the difference between what he calls the *product view* of language, encompassing studies of *linguistic competence*, *language structure*, and *semantics*, with the *action view* of language, encompassing studies of *linguistic performance*, *language use*, and *pragmatics*. He gives three tenets for the study of language use:

- In language use, utterances are more basic than sentences.
- In language use, speaker's meaning is primary, and word or sentence meaning are derivative.
- Speaking and listening aren't autonomous activities, but parts of collective activities.

The action view begins with Austin's observation that utterances in conversation are *speech acts*, and as such should be treated as part of a theory of action [Austin, 1962]. This observation and the subsequent research program on speech acts within the philosophy of language has been followed up on by researchers in AI, treating speech acts in the same manner as other actions that an agent can perform, recognize another agent performing, and reason about. The traditional approach has been to view speech acts as changes to the *cognitive state* of another agent, analogous to the way physical actions change the state of the physical world. Speech act operators have been devised using the formalisms from AI planning systems, so that deciding what to say can be seen as *utterance planning*, and interpretation of the intention behind an utterance can be viewed as *plan recognition*.

Given that utterances are actions, what kinds of actions are they? What are the enablement and generation conditions and what are the effects? One difficulty with formalizing speech acts in this way is that, as Clark's third tenet claims, all speech acts are collaborative acts: they require both a speaker to make an utterance and a listener to understand and accept it in some way. The result of a speech act is thus, in some ways, negotiated by the conversational participants. A successful formulation of speech acts will have to be based on a theory of *multi-agent action*.

The collaboration process is further complicated by the fact that participants cannot infallibly recognize the mental states of the other participants: the hearer of an utterance cannot know for sure that he has understood the intent of the speaker (speaker meaning),

and knowing this fact about the hearer, the speaker cannot be sure that she has been understood. In general, many of the effects of speech actions will be be to the cognitive states of conversational participants. Because each agent will not have direct access to the cognitive states of other agents, effects of speech acts and the intentions behind them are intrinsically uncertain.

Most previous natural language understanding systems have largely ignored the problem of coordinating understanding and have assumed that the intent of the utterance can be recognized merely by observing the occurrence of the utterance, using only the form of the utterance itself plus background context including knowledge of the other participant. Since most of these systems have been built as part of a *question-answering* system, with system responses resulting from a single database retrieval and in which complicated discourse interactions aren't possible, or as part of a *story understanding* system, in which there is no facility for interaction and off-line processing can be performed at leisure.

In contrast, the study of conversation shows that there is quite a rich system for coordinating understanding. For example, studies of conversations in the TRAINS domain[1] show that about half of the utterances in a conversation are related to coordinating understanding rather than being domain-level utterances on the topic of the conversation [Allen and Schubert, 1991]. There is an acknowledgement system to make sure that utterances are heard and understood. There is an acceptance system so that proposals and information can be agreed on. There are facilities for clarification and repair of potential or actual misunderstandings. These facilities have been the object of extensive study in the field of Conversation Analysis, but have only recently been adopted in AI conversation systems (see Section 2.4).

A difficulty in formulating descriptions of actions (such as speech acts) in a multi-agent setting is determining what point of view is being described. Three common points of view are:

1. The objective (what "really" is) (sentence meaning)
2. The point of view of the performing agent (speaker meaning)
3. The point of view of an observing agent

Ideally, one would like all three to coincide, i.e., the actor decides what she wants to do, performs an action which accomplishes this intention, and the intention and action are correctly recognized by the observer. Unfortunately, as mentioned above, this kind of situation is not guaranteed. The actor may have incorrect beliefs, or may fail in her action, so that what she believes she did is not what she "really" did. The observer also has limited knowledge, and may misinterpret an action. He also has no access to and only limited evidence about the mental state of the actor, and may not recognize what is intended.

For linguistic communication, what "really" happened is of less importance than that the conversing agents reach some sort of understanding. The question of whether

[1] Described below in Section 4.3.

the meaning of an utterance has some objective status aside from what is intended and recognized by the agents is controversial and not really relevant here. All that is important for communication is that one agent used a particular locution to convey some content to another agent, and that the speaker's intention to convey the content becomes mutually understood (*grounded*) by both agents, regardless of any objective meaning of the utterance.

These distinctions have not generally been made in most speech act work, so it is often difficult to tell the ontological status of many proposed acts: are they objective phenomena, observable and testable by an (ideal) observer? Are they part of the mental states of the agents, consciously used and necessary for getting at what was intended? Is a set of speech acts to be interpreted as an objective description of the conversation process, or a psychological model of communicating agents (or both)?

A complete theory of action in conversation is a project far too ambitious to be undertaken in one dissertation. Instead, I pick out one important subproblem, the *grounding* problem, and show how an action-oriented approach can shed light on its difficulties. I approach this problem from several different angles: how best to characterize particular utterance types as actions, how to use this theory of action within a computer system which converses in natural language, and how to formally characterize conversational actions within a more general logic of action. For each of these problems, I present the skeleton of a more general theory, and show in detail how the grounding problem can be approached within this framework.

1.1 The Grounding Problem

It is uncontroversial that successful communication requires some degree of common ground between the communicators. In order to communicate in a natural language, for instance, the communicators must have some agreement on the meanings of words, the legal combinations of words, and the meanings of these combinations. Not only must a speaker know these, she must assume that her hearer knows them and knows that she knows them, etc. This type of common ground is often called mutual knowledge, mutual belief, shared information, and other similar terms. In addition to this basic type of common ground, researchers have used concepts like mutual belief as components in theories of a wide range of social and linguistic phenomena, including *linguistic meaning* [Schiffer, 1972], *definite reference* [Clark and Marshall, 1981], *conventions* [Lewis, 1969], *social norms* [Bach and Harnish, 1979], and *coordinated activity* (many authors including [Levesque *et al.*, 1990]).

An important question is how this common ground is augmented by engaging in conversation. This process is termed *Grounding* by Clark and Schaefer [1989]. The famous Byzantine generals problem illustrates that actual mutual knowledge cannot be achieved in a situation in which communication is fallible [Halpern and Moses, 1990]. Yet, for the most part, conversants seem to get by well enough. Clark and Schaefer suggest that such mutuality need not be absolute, but merely sufficient for the current purposes.

1.2 Thesis Statement

We provide a computational model of how conversants reach a state of mutual understanding of what was intended by the speaker of an utterance. Most previous computational NL systems and AI theories have ignored the problem, assuming that this happens more or less automatically as a result of the speaker and hearer being together, and instead have concentrated on the problem of correctly interpreting the meaning of the utterance. Instead we say that it is less important to come up with the "correct" interpretation than to get an approximately correct interpretation, and then use repairs to patch things when the interpretation is not close enough for current purposes (the *grounding criterion* of Clark and Schaefer [1989]). This approach seems increasingly necessary as researchers are finding many important problems in natural language processing and planning which are intractable for the optimal case [Perrault, 1984; Chapman, 1987; Reiter, 1990]. Several researchers in other fields have come up with similar schemes for presenting a post-hoc analysis of grounding in conversation, but have not presented formal models which would show how an agent could achieve common ground on-line.

1.3 Grounding in Human and Computer Communication

The grounding problem occurs not only for natural language communication, but any time two agents need to coordinate on transfer of information. A simple example is one of the computer communication protocols. These protocols not only allow transmission of information (in data packets called information frames, or *I-frames*, from a sender to a receiver), but allow the receiver to send back acknowledgements and negative acknowledgements, as well as allowing the sender to repair transmission errors. Most protocols use a single acknowledgement scheme in which the sender assumes the data was correctly received if an acknowledgement is received and the receiver assumes the sender believes the data was correctly received when the receiver sends the acknowledgement. If the sender does not receive an acknowledgement, then it resends the packet. Likewise, if the sender gets a negative acknowledgement (signaling a corrupted or missing packet) it can resend the packet.

We can say that the communicating systems have grounded a particular I-frame when an acknowledgement for it is received (the receiver of the original I-frame assumes the packet is grounded unless/until it gets a retransmission). There are many similarities between these protocols and grounding in natural language conversation. For instance, both incorporate different styles of acknowledgement: in computer communication, one style of acknowledgement, called *echo-checking*, is to have the responder repeat back the original message. This also happens frequently in human conversation, as in the exchange in (1).

(1) S: It would get there at 4.
M: It would get there at 4.

For both types of communication, this is a fairly expensive style of acknowledgement and is used primarily in cases where it is very important to demonstrate that the receiver got the message exactly right and when there is plenty of time. Another style of acknowledgement is to send a separate message type, an *Ack* message. In English, the following utterances typically have the meaning that the previous utterances were understood: "uh-huh", "yeah", "right", "okay."

There is also a third type of acknowledgement common to both media. In computer communication in which information is being transmitted both ways, some protocols allow an alternative to sending a separate *Ack* message. This is to *piggyback* the acknowledgement onto the next I-frame sent by the receiver of the initial I-frame. There is a similar strategy in natural language conversation, in which a message will implicitly acknowledge a previous message, when the acknowledging message would not be sensible without the initial message. A simple example of this type of phenomena is an answer to a question, as in (2).

(2) M: How long does it take to load oranges?
S: An hour.

Another example would be a follow-up response such as (3).

(3) M: Let's ship the oranges.
S: And we'll have E3 pick them up.

NL conversation also has a rich supply of forms for negative acknowledgement, forms which specify lack of receipt of a clear message, such as, in English, "What?", "Pardon?", and "Please repeat."

There are, however a number of differences between NL conversation and computer communication. First of all, in NL conversation, rather than having a fixed "frame" size, utterances can be of varying lengths, with wide-ranging information contents. Also, unlike data communications where I-frames have ID numbers, in NL conversation there is generally no way of identifying a particular utterance, or of linking a general acknowledgement to a specific utterance.

Furthermore, grounding in NL conversation generally occurs at the level of meaning level, rather than form. In most cases, it is perfectly reasonable to acknowledge an utterance in which the gist of the utterance is understood, even though the precise form was not heard correctly. Also, if the form was heard correctly, but is not comprehensible in the utterance situation, this is a situation for repair. In contrast, most data communication is acknowledged if and only if the actual form was received correctly, regardless of whether the receiver can fathom the meaning of what was sent.

Finally, natural language has a much more flexible system of repair. Rather than just repeating a previous utterance, a speaker can repair only the problematic parts of an utterance, as in (4), In which the speaker changes the question from one of tanker contents to tanker location.

(4) S: Do you have to have the tanker full?
: uh there?

Also, in addition to asking for a retransmission or not responding, the listener of an utterance can also repair or request repair of a specific part of the utterance, as in (5).

(5) S: and then which route do you wanna take?
M: took to Avon?
S: yeah

These increases in flexibility come at a cost of precision. It is thus harder to determine with a given level of certainty that a particular utterance was understood correctly. Neutral acknowledgements like "uh-huh" signal, at best, a claim that the previous utterances were understood, without being able to specify that, say, utterances 1, 2, and 4 were understood while 3 was never received.

The challenge, then could be phrased as follows: is there a protocol in natural language conversation for grounding? What is the form of this protocol, and how can it be used by a computer system to allow it to engage in natural conversation? What are the message types, how are they recognized, and how do transmission and receipt of these messages affect the states of the participating agents?

1.4 Contributions of the Dissertation

This dissertation meets the challenge from several directions. First, a set of abstract actions for grounding is induced from study of task-oriented dialogues. These actions are somewhat orthogonal to traditional speech acts concerned with illocutionary force and propositional content. They take place at an *utterance* level rather than sentence level, and are concerned with how the utterance affects the grounding process. This level of action is shown to fit coherently within a multi-level model of action in conversation including also levels for traditional speech acts as well as turn-taking acts and higher-level discourse coherence acts.

A protocol is designed which can determine, for any sequence of grounding acts, whether the content expressed by the utterances which comprise the acts is grounded or not. This protocol is also compared to mental models of language participants to provide rationales for taking particular actions. The protocol is then implemented within a NL conversation system, allowing the system to reason about the state of grounding and perform appropriate acknowledgements and repairs. Finally, the grounding acts used in the protocol are given precise definitions within the context of a general theory of plan execution.

1.5 Outline of Dissertation

Chapter 2 gives an overview of some of the previous research programs on which the current work builds. It also presents a more detailed analysis of the complexities of speech acts, user models, multi-agent interaction, action in conversation, and grounding.

Chapter 3 presents the framework of the grounding theory, providing a protocol for grounding in conversation, giving its rationale for construction, and its relation to discourse context and the mental state of the conversational participants.

Chapter 4 describes the theory of *conversation acts*, a multi-stratal theory of action in conversation, and shows how grounding fits in as a *level* of conversation acts, along with levels for traditional speech acts, turn-taking acts, and argumentation acts.

Chapter 5 describes an architecture for dialogue management that can be used to control the functioning of a natural language interface. This architecture includes a user model and use of conversation acts (including grounding acts) to update this model.

Chapter 6 describes the implementation and functioning of these ideas within the TRAINS Conversation System. Included is a trace of the system running on a sample dialogue, which is concluded in Appendix A.

Chapter 7 presents a formal theory of plan execution which is able to represent repair and sub-action. This theory is then used to formalize the grounding acts from Chapter 4.

Chapter 8 summarizes the contributions made in this dissertation and describes some natural extensions to the work described in the previous chapters.

2 Foundational Work

This chapter presents an overview of some of the previous research programs on which the work in the following chapters builds. It also presents an analysis of the complexities of speech acts, multi-agent interaction, action in conversation, and grounding. Section 2.1 describes some of the foundational work from both philosophy of language and AI in defining speech acts and formalizing them as planning operators in computational systems. Section 2.2 summarizes some of the work on modelling the mental states of agents, particularly belief, mutual belief, intentions and plans, joint intentions, and obligations. Most importantly for grounding is the problem of representation and acquisition of mutual belief between agents, which is also generally taken as one of the main intended effects of speech acts. Section 2.3 relates some of the most important insights from the subfield of Sociology known as *Conversation Analysis* which are relevant for the present topic. Section 2.4 describes previous attempts to incorporate ideas from Conversation Analysis into natural language processing systems. Section 2.5 examines the proposals put forth by Clark and his colleagues for a descriptive model of grounding. Finally, Section 2.6 presents some deficiencies with Clark's model of grounding which must be overcome to provide a protocol for grounding.

2.1 Speech Acts and Plans

2.1.1 Foundational Philosophical and Linguistic Speech Act Work

Austin

Austin observed that utterances are not just descriptions of states of affairs, but are used to *do* things [Austin, 1962]. Under felicitous circumstances, utterances can change the mental and interactional state of the participants. Speaking is *acting*, and in speaking the speakers are producing *speech acts*. There are multiple types of action performed in speaking, and Austin distinguished several. *Locutionary acts* are the act of saying something, including a *phonetic act* – producing certain noises, a *phatic act* – producing words belonging to a vocabulary in a constructions conforming to a grammar, and a *rhetic act* – using the product of the phatic act with a particular sense and reference.

Illocutionary acts are those acts performed *in* saying something, for example, asking or answering a question, giving some information, etc. Most of the subsequent work on

speech acts has been on illocutionary acts. Illocutionary acts are taken to be composed of an *illocutionary force*, which specifies the type of action (e.g., requesting, suggesting, warning, apologizing, informing), and a *propositional content* which specifies the details of the action (e.g., what it is that the hearer is being requested to do). Illocutionary acts are not always directly deducible from the locutionary acts which generate them. *Indirect speech acts* are those in which the act performed is other than what would be expected from a compositional account of the content. Austin gives a bridge example in which an utterance of "I bid three clubs" is used to inform a partner that the speaker has no diamonds.

Perlocutionary acts are those which are performed *by* saying something – actions which achieve effects which are special to the particular situation of utterance rather than the conventional nature of the communication. Examples include persuasion, surprise, and deterrence.

Searle

Searle extends and refines Austin's work on illocutionary acts. He observes that Austin's decomposition of speech acts into *illocutionary force* and *propositional content* shows up in the different kinds of negation that can be performed in a sentence [Searle, 1969]. For example, the sentence "I promise to come" has two negations: a *propositional negation*, "I promise not to come", in which the illocutionary act (promise) is the same, but in which the content is negated, and "I do not promise to come", in which the propositional content is the same, but the illocutionary act is no longer a promise, but a *refusal* to make a promise. Searle further decomposes the *propositional content* of an act as a combination of *predicating* and *reference* acts. Neither of these stand alone but are performed only in concert with illocutionary acts.

Probably the most important contribution was an attempt to provide necessary and sufficient conditions for the performance of illocutionary acts. He presented these as *constitutive rules* (like the rules which define the games of football or chess) of various sorts. *Normal input-output conditions* concern the conditions of intelligible speaking and understanding, including knowing the conventions of languages, paying attention, etc. *Propositional content conditions* describe restrictions on the content, e.g., for a promise the content must be a future action. *Preparatory conditions* involve the constraints on the world that make the speech act useful. *Sincerity conditions* involve alignment of the speaker's actual attitudes (belief, desire, etc.) with the attitudes expressed by the act. *Essential conditions* involve the speaker's intentions in performing the act – what she was *trying* to do. Searle also adds a condition based on Grice's notion of non-natural meaning [Grice, 1957], that the effect of the act is in part produced by the hearers recognition that the utterance is intended to produce this effect by means of the recognition of the intention.

Searle also improves Austin's classification of types of illocutionary acts [Searle, 1976]. Austin's classification (into *verdictives, exercitives, commissives, expositives,* and *behavitives*) is fairly haphazard, based more on similarity of illocutionary verbs than the acts themselves. Searle proposes an alternate taxonomy based on the purposes

of the acts. Searle's taxonomy includes: *representatives*, which commit the speaker to the truth of an expressed proposition, *directives*, which involve getting the hearer to do something, *commissives*, which involve committing the speaker to some course of action, and *expressives*, which convey a psychological state of the speaker.

2.1.2 Foundational AI Speech Act Work

Bruce

Bruce was the first to try to account for Speech Act theory in terms of AI work on actions and plans [Bruce, 1975]. He defined natural language generation as *social action*, where a *social action* is one which is defined in terms of beliefs, wants, and intentions. He also presented *Social Action Paradigms* which showed how speech acts could be combined to form larger discourse goals. He showed how acts such as Inform or **Request** could be used in achieving intentions to change states of belief.

Allen, Cohen, and Perrault

Cohen and Perrault [1979] defined speech acts as plan operators which affect the beliefs of the speaker and hearer. They set up write that any account of speech acts should answer the following questions:

- Under what circumstances can an observer believe that a speaker has sincerely and successfully performed a particular speech act in producing an utterance for a hearer?
- What changes does the successful performance of a speech act make to the speaker's model of the hearer, and to the hearer's model of the speaker?
- How is the meaning (sense/reference) of an utterance x related to the acts that can be performed in uttering x?

They also suggest that a theory of speech acts based on plans should specify the following:

- A planning system: a formal language for describing states of the world, a language for describing operators, a set of plan construction inferences, a specification of legal plan structures. Semantics for the formal languages should also be given.
- Definitions of speech acts as operators in the planning system. What are their effects? When are they applicable? How can they be realized in words?

These issues are still central to the work going on in discourse planning.

Cohen and Perrault's models of mental states consist of two types of structures: *beliefs* and *wants*. *Beliefs* are modal operators which take two arguments: an agent who is the believer, and a proposition which is believed. They also follow Hintikka [1962],

augmenting the belief structure to include quantified propositions. Thus an agent can believe that something has a value without knowing what that value is, or an agent can believe another agent knows whether a proposition is true, without the first agent knowing if it's true or not. *Wants* are different modal operators which can nest with beliefs. *Wants* model the goals of agents.

Perrault and Cohen then proceeded to make a first stab at satisfying these issues. The planning system they use is a modified version of STRIPS [Fikes and Nilsson, 1971]. They maintain STRIPS's method of dealing with the frame problem: assuming that nothing can change the world except the explicit changes mentioned by the effects of an operator. They describe two different types of preconditions, both of which must hold for the action to succeed. *cando* preconditions indicate propositions which must be true for the operator to be applicable. *Want* preconditions are meant to cover sincerity conditions. In order to successfully perform an action, the agent (speaker) must want to do that action. They model the speech acts REQUEST and INFORM, within their planning system.

Allen and Perrault [1980] use essentially the same formalism as Cohen and Perrault, but for a slightly different purpose. They investigate the role of plan inference and recognition in a cooperative setting. They show how the techniques of recognizing another agent's plans can allow one to recognize an indirect speech act in a coherent and relevant manner. The planning system is again, basically a STRIPS system. There are preconditions and effects, and a body, which is a specification of the operator at a more detailed level.

Litman and Allen

Litman and Allen extend Allen and Perrault's work to include dialogues rather than just single utterances, and to have a hierarchy of plans rather than just a single plan [Litman, 1985; Litman and Allen, 1990]. They describe two different types of plans: domain plans and discourse plans. Domain plans are those used to perform a cooperative task, while discourse plans, such as *clarification* and *correction*, are task-independent plans which are concerned with using the discourse to further the goals of plans higher up in the intentional structure. They also use a notion of *meta-plan* to describe plans (including discourse plans) which have other plans as parameters. Using these notions, Litman and Allen are able to account for a larger range of utterances than previous plan-based approaches, including sub-dialogues to clarify or correct deficiencies in a plan under discussion. There is still no facility for explaining acknowledgment, as the assumption of perfect understanding is maintained.

Non-monotonic Theories of Speech Acts

Perrault [1990] takes as a starting point the problem that the utterance itself is insufficient to determine the effects of a speech act. All effects of utterance actions are based in part on the prior mental states of the agents as well as what was actually uttered. However, formalizing the precise conditions which must hold is a tricky endeavor, because

of the many possible contingencies. Thus an axiom stating the effects of an utterance in declarative mood must take account of the possibilities of lies, failed lies, and irony as well as standard information-giving acts. Perrault's approach is to state the effects in terms of Default Logic [Reiter, 1980], so that the simple, most common effects can be derived directly, unless there is some defeater. He has a simple axiomatization of belief, intention and action, along with some normal default rules, including a *Belief Transfer* rule which says that if one agent believes that another agent believes something the first agent will come to believe it too, and a *Declarative* rule, which states that if an agent said a declarative utterance, then it believes the propositional content of that utterance. This simple schema allows Perrault to derive expected consequences for the performance of a declarative utterance in different contexts.

Although the formalization is simple and elegant, it still contains a number of serious difficulties. Foremost is the lack of a serious treatment of belief revision. Although intuitively, speech acts are used to *change* beliefs, Perrault's framework can only handle the case of new beliefs being added. As well as not allowing the kind of discourses in which one agent would try to change the beliefs of another, the logic also has the strange property that one agent can convince itself of anything it has no prior beliefs about merely by making an utterance to that effect in the presence of another agent! The logic also does not lend itself to a computational implementation, since one would need a complete, inductive proof scheme to make all of the necessary deductions.

Appelt and Konolige [1988] reformulate Perrault's theory in terms of Hierarchic Autoepistemic Logic [Konolige, 1988]. This reformulation has the advantages of implementability and the ability to order the defaults to overcome the problems that Perrault had with normal default logic, but it also loses the simplicity of Perrault's framework. It is hard to see whether Appelt and Konolige are trying to describe something from the point of view of an ideal observer or from a participant in the conversation. In formulating their theory, they also resort to some unintuitive devices such as the beliefs of an utterance.

Cohen and Levesque

Cohen and Levesque have been attempting to solve a number of problems relating to formal characterizations of Speech Acts, through the use of a logic of action and mental attitudes. [Cohen and Levesque, 1990b] lays out the framework of the basic theory of rational action. It is based on a dynamic modal logic with a possible worlds semantics. They give axiomatizations for modal operators of beliefs and goals, and then derive intentions as persistent goals, those to which an agent is committed to either bring about or realize are unachievable.

[Cohen and Levesque, 1990c] uses this logic to show how the effects of illocutionary acts can be derived from general principles of rational cooperative interaction. They claim, contrary to [Searle and Vanderveken, 1985], that communicative acts are not primitive. They define what it means for an agent to be sincere and helpful, and give characterizations of imperatives and requests. They claim that recognizing the illocutionary force of an utterance is not necessary, that all that is important is that the

hearer do what the speaker want, not that he recognize which act the speaker performed as a part of this process. They thus appear to be claiming that illocutionary acts should be seen as descriptive models of action, not as resources for agents. They conclude with a description of how Searle and Vanderveken's conditions on acts can be derived from their rational agent logic.

[Cohen and Levesque, 1990a] extends the framework to handle Performatives. They define all illocutionary acts as attempts. Performatives are acts which have a request component and an assertion component, and the assertion component is made true merely by the *attempt*, not the success of the action. Thus *request* is a performative verb, while *frighten* is not (because it requires a successful attempt and the success is beyond the control of the speaker), and *lie* is paradoxical when used performatively, because the explicit mention defeats the aim.

[Cohen and Levesque, 1991a] present an analysis of why confirmations appear in task-oriented dialogue. Using their theory of joint intentions developed in [Levesque *et al.*, 1990] (described below in Section 2.2.4), they state that the participants in one of these task-oriented dialogues have a joint intention that the task be completed. It is part of the definition of joint intention that if one party believes the object of intention to be already achieved or to be unachievable, that party must strive to make the belief mutual. It is this goal of mutual belief which drives the agent to communicate a confirmation. Although this is perhaps the first attempt in the computational literature which is explicitly concerned with a plan-based account of the generation of confirmations, it is noticeably lacking in several respects. It has no mention of how the intention to make something mutually believed turns into an intention to perform a confirmation. There is also some distance still from the logic to actual utterances. It is not explained just what would count as a confirmation, and how one might recognize one.

Cohen and Levesque have provided a nice formal logic with which to precisely state and analyze problems of multi-agent coordination and communication, but it is difficult to see how it could be used by a resource-bounded agent in planning its actions or recognizing the intentions of others.

2.1.3 Multi-Agent Planning

A speech act theory which can account for conversations must include at least the following extensions to classical planning (e.g. STRIPS):

- temporal reasoning, including reasoning about overlapping and simultaneous actions
- uncertainty: attempted actions may fail to achieve their desired results, unexpected results may follow.
- multiple agents, each with individual knowledge, goals, etc.
- cooperation among agents
- real-time resource-bounded reasoning

- integration of planning and acting

There is a large amount of research dedicated to addressing these problems, much more than can be summarized here. [Traum and Allen, 1991] explores some of the complexities involved in reasoning and acting in a richer environment. The annual European workshops on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW) (reprinted in [Demazeau and Muller, 1990; Demazeau and Muller, 1991; Werner and Demazeau, 1992]) contain a variety of approaches to these problems.

2.2 Models of Mental States

One of the key features of *speech acts* as opposed to physical actions is that their main effects are on the mental and interactional states of agents, rather than on the state of some external domain. This section relates some of the most relevant work on representing and reasoning about several of those mental states, including belief, intentions and plans, mutual belief, shared plans, and obligations. This work serves as a background for the theory of *Conversation Acts* presented in Chapter 4 and a basis for the dialogue manager implementation in Chapter 5.

2.2.1 Belief

Belief and *Knowledge* are the attitudes that have been most studied in both the philosophical and AI literatures. Although in informal language, the word "knowledge" is often used to mean a strong belief, it has acquired a technical meaning which includes both belief and truth (often among other conditions). Since, in realistic situations, an agent does not generally have access to the actual state of affairs, we will generally refer to *belief*, though this is also the attitude generally modelled by *knowledge bases*. The literature on belief modelling is far too vast to attempt even a cursory summary, but we will mention some of the more influential approaches and some of their defects as models of actual belief.

The beliefs of an agent is that agent's own model of how things are. In representing the beliefs of an agent, it is important not only to represent explicit facts that the agent is aware of, but also what else these facts suggest to the agent about the world. This usually managed by allowing some inference rules to augment explicit beliefs with other natural consequences. A predicate in simple first-order logic is not powerful enough to serve as a representation of belief, since the object of belief is not a set of simple terms, but is a formula itself. The usual solution, first proposed by Hintikka [1962], is to use a modal logic for belief. Hintikka later presented a semantics for belief based on accessibility functions over possible worlds [Hintikka, 1971]. A proposition is believed by an agent if that proposition is true of all the worlds in the accessibility relationship for that agent.

While this framework has been very influential, it has some serious drawbacks as an actual model of human belief. For one thing, it predicts that all logical consequences of beliefs will also be believed. While, in general, this can be a convenient facility, it is far

too strong. Unfortunately, there have yet to be developed adequate restrictions on this type of inference so that just the "correct" consequences will be modelled.

2.2.2 Intention

Intention is an attitude relating an agent to actions that she has *decided* to perform. Intentions also play a causative role in the occurrence of the actions and serve also to constrain the further deliberation of the agent. Bratman presents probably the most detailed theory of intention and its relation to other attitudes and processes [Bratman, 1987; Bratman, 1990]. Cohen and Levesque present a theory of intention based on more primitive *commitments* [Cohen and Levesque, 1990b]. Konolige and Pollack present a formalization of intentions within a non-normal modal logic [Konolige and Pollack, 1993].

Intentions are also important in utterance interpretation and plan recognition in general. Many analyses of utterances are concerned not just with what happened, but with what the speaker *intended* to do in performing the action. Also important is the structuring of individual intentions into *plans* and the *Intentional Structure* of Discourse [Grosz and Sidner, 1986] – how the intentions behind utterances in a discourse are related.

2.2.3 Mutual Belief

Most of the theories of speech acts as plans reported in Section 2.1 have as some of the main effects of speech acts the addition of some new *mutual beliefs*. Mutual beliefs are also taken to be some of the prerequisites for felicitous utterance of speech acts. But just what are mutual beliefs? This section reviews some of the proposals for how to represent the properties of mutual beliefs in terms of simpler beliefs, and how one could acquire new mutual beliefs. As with *belief* and *knowledge*, there have been a variety of names for a cluster of related concepts, including "mutual knowledge", "common knowledge", "mutual belief". In the discussion below, we generally use the term used by the author under discussion, but treat these terms as synonymously meaning *mutual belief*, in which what is mutually believed by a group of agents is not necessarily actually true.

Formulations of Mutual Belief

While people agree for the most part about the intuitions underlying the phenomenon of mutual belief, there have been a variety of different ways proposed of modeling it. Barwise [1989] compares model theories for three different formulations.

Schiffer uses what Barwise calls "the *iterate* approach" [Barwise, 1989, p. 202]. He defines mutual knowledge between two agents A and S of a proposition p, $K^*_{SA}p$ as [Schiffer, 1972, p. 30]: $K_Sp \wedge K_Ap \wedge K_SK_Ap \wedge K_AK_Sp \wedge K_SK_AK_Sp \wedge K_AK_SK_Ap \wedge \cdots$ It is thus an infinite conjunction of nested beliefs. This approach has since been adopted by many others, including Allen [1983b] and Perrault, who provides an elegant default

logic theory of how to obtain each of these beliefs given prior knowledge and a conversational setting [Perrault, 1990].

Barwise credits Harman with the *fixed-point approach.* Harman formulates mutual knowledge as "knowledge of a self-referential fact: A group of people have mutual knowledge of p if each knows p *and we know this*, where *this* refers to the whole fact known" [Harman, 1977, p. 422]. As Barwise points out, the fixed-point approach is strictly stronger than the iterate approach, because it includes as well the information that the common knowledge is itself common knowledge. It also replaces an infinite conjunction with a self-referential one.

The final approach discussed by Barwise is the *shared-situation approach.* He credits it to Lewis. Lewis formulates rules for common knowledge as follows [Lewis, 1969, p. 56]:

> Let us say that it is *common knowledge* in a population **P** that **X** if and only if some state of affairs **A** holds such that:
>
> 1. Everyone in **P** has reason to believe that **A** holds.
> 2. **A** indicates to everyone in **P** that everyone in **P** has reason to believe that **A** holds.
> 3. **A** indicates to everyone in **P** that **X**.

This schema is also used by Clark and Marshall, and is apparently the one which Barwise himself endorses.

[Cohen, 1978] uses a *belief spaces* approach to model belief. Each space contains a set of propositions believed by an agent. Nested belief is represented by nested spaces. There is a space for the system's beliefs (SB) which can contain a space for the system's beliefs about the user's beliefs (SBUB) which in turn can contain a space for the system's beliefs about the user's beliefs about the system's beliefs (SBUBSB). If Cohen were to adopt the iterated approach directly, it would require an infinity of belief spaces. Instead, he takes the space one deeper than the deepest which contains any non-mutual beliefs, and points it to its parent space, thus creating a loop, where each even nesting is the same as every other even nesting. Now each of the nested beliefs in the iterated approach can be generated or seen to be present in his belief spaces, by iterating through the loop. This approach shares some features with the fixed-point approach (the self-referentiality) and it allows quick determination of whether mutual belief exists (by searching for a loop), unlike the iterated approach, but it is in fact not as strong as the fixed-point approach because the higher-order implications of the fixed-point approach, such as mutual belief about the mutual belief, cannot be represented.

A slight modification is to add a separate kind of space, a *mutual belief* space to represent mutual beliefs. This is the approach taken by Bruce and Newman [1978]. The Rhetorical knowledge representation system [Allen and Miller, 1991] also uses a mutual belief space, but disallows nested beliefs within a mutual belief space, giving essentially the power of Cohen's system. This also seems to be the approach used by Maida [1984].

How can Mutual Belief be Achieved?

If Mutual Belief includes at least the infinite conjunction of nested beliefs, there is a problem as to how to achieve mutual belief, or to recognize when it has been achieved. Several researchers have put forth proposals, as described below, yet none seem completely satisfactory.

Perrault uses an extremely strong set of assumptions to drive his default theory [Perrault, 1990]. He has an axiom of observability which states that if an agent is "observing" another agent, then he will recognize all actions (such as declaring a certain proposition) performed by that agent. Agents also have complete memory of prior beliefs, and persist their beliefs into the future (Perrault's theory can not handle belief revision). He also has two default rules, a *belief transfer* rule which states that if one agent believes that a second agent believes something, then the first agent should come to believe it (assuming it doesn't conflict with his prior beliefs), and a *declarative rule* which states that if an agent declares a proposition, then he believes it to be true. With Perrault's set-up, one can derive all the nested beliefs of the iterated approach, assuming there were no prior contradictory beliefs. In the case of some prior inconsistent beliefs, such as in the case of a lie or ironic assertion it also derives the correct set of beliefs. However, from a computational perspective it is difficult to see how an agent using Perrault's framework could recognize mutual belief without an infinite amount of computation (or at least some kind of inductive proof procedure for default logic). Perrault also doesn't mention what might happen in the case where his assumptions are too strong.

Clark and Marshall [1981] describe two kinds of heuristics to get at mutual knowledge in a finite amount of time. *Truncation heuristics* look at just a few of the nested beliefs, and then infer mutual belief if all of those check out. *Copresence heuristics* involve the agents recognizing that they and the object of mutual knowledge are jointly present. Clark and Marshall discount the truncation heuristics as implausible, since it is hard for people to reason overtly about nested beliefs. Also, the situation that usually provides evidence for the beliefs checked by the truncation heuristic is usually what would be used directly by the copresence heuristics.

They list four main ways of achieving the copresence necessary for mutual belief, with subdivisions of some of these. Their table with the auxiliary assumptions [Clark and Marshall, 1981, p. 43] is reproduced as Table 2.1.

Community co-membership is achieved when two agents mutually know that they are part of some community (e.g. people, squash players, computer scientists, etc.). *Universality of knowledge* refers to the assumption that certain things will be mutually known by everyone in a community. These two assumptions together, that two agents **A** and **B** are part of a community and that everyone in this community mutually knows **x**, are sufficient to conclude that **A** and **B** mutually know **x**.

The *simultaneity* assumption is that the agents are simultaneously in the same situation. The *attention* assumption is that the agents are paying attention to the shared situation. The *rationality* assumption is that the agents are rational, and can draw normal inferences. If the situation is a case of physical copresence, then if it is a case

Basis for mutual knowledge	Auxiliary assumptions
1. Community membership	Community co-membership, universality of knowledge
2. Physical copresence	
a. Immediate	Simultaneity, attention, rationality
b. Potential	Simultaneity, attention, rationality, locatability
c. Prior	Simultaneity, attention, rationality, recallability
3. Linguistic copresence	
a. Potential	Simultaneity, attention, rationality, locatability, understandability
b. Prior	Simultaneity, attention, rationality, recallability understandability
4. Indirect copresence	
a. Physical	Simultaneity, attention, rationality (locatability or recallability), associativity
b. Linguistic	Simultaneity, attention, rationality, (locatability or recallability), associativity understandability

Table 2.1: Clark & Marshall's Methods of Achieving Copresence for Mutual Knowledge

of immediate copresence these three assumptions are sufficient, (e.g. Ann and Bob are looking at a candle, and looking at each other looking at the candle, so a definite reference of *the candle* is felicitous). If the situation is in the past, then an additional assumption of *recallability* is necessary: it's not enough that the situation occurred, they have to remember it. If the situation hasn't happened, but very easily could, then *locatability* is needed. For example, say Ann and Bob are in a room with the candle, but not looking at it; then a reference is felicitous, assuming that the candle is locatable: the reference itself would provide the impetus for achieving the shared situation.

For linguistic copresence (reference to an object mentioned in prior discourse) an additional assumption is required, *understandability*. This is that the utterance which introduces the object can be understood as having done so. The final type of mutual knowledge is a mixture of community co-membership and one of the other two. An example of this is when a candle has been introduced, and then a definite reference to the price, or the wick is made. An additional assumption of *associativity* is needed to be sure that the hearer can make the connection.

There is still a problem with Clark and Marshall's characterization, in fact, the same problem which motivated them to take up mutual knowledge in the first place. Their conditions for potential copresence are not sufficient. Taking the example they use to show the insufficiency of any finite set of nested beliefs for definite reference, we can see it is also insufficient for potential coreference. Assuming a prior episode of Ann and Bob looking in the morning newspaper and seeing that *A Day at the Races* is playing at the Roxy theater, if Ann, later sees a correction in the evening paper that the movie will be

Monkey Business, it would not be a felicitous reference to say "the movie at the Roxy" to mean *Monkey Business*. This is true even if Bob has seen the correction, and Ann knows Bob has seen the correction, and she knows he knows she knows he has seen the correction. As long as the sequence is finite, the chain always bottoms out, and we are left with *A Day at the Races* being the more felicitous, according to Clark and Marshall. But this is precisely the situation with potential coreference. In normal circumstances, Ann can ask if Bob has seen the movie even if she doesn't know if he knows what it is, as long as he can locate what the movie is – perhaps the paper is in front of him. But if we have the prior circumstance of joint knowledge of another referent, we have a problem, no matter how locatable the intended referent is. There are several difficulties in using Clark and Marshall's account: we must not only pick out the unique object in the situation which the definite description refers to, we must also pick out the (unique?) situation in which we can find such an object. This suggests first of all that Clark and Marshall's assumptions for potential copresence are insufficient, and secondly, that perhaps, as Johnson-Laird suggests [1982, p. 41], their examples do not show that mutual knowledge is necessary. Clark and Carlson [1982, p. 56] counter that they are talking about mutual expectation and belief as much as knowledge, and thus Johnson-Laird's proposed counterexamples are not problems for their account. There is still the following potential difficulty: as shown above, the assumptions for potential copresence are not sufficient; therefore, something else is needed. Perhaps this something else can also get them out of the original problem without recourse to mutual knowledge. To use their formulation, one must at least work out the relationships between different basis situations, and provide a method of choosing which of competing assumptions should be made.

Clark and Marshall recognize that reference can fail and can be repaired. They distinguish two types of repair, which they term *horizontal repair* and *vertical repair*. *Horizontal repair* refers to giving more information about the item, but keeping the basis (the type of copresence) the same, whereas *vertical repair* is giving a new basis (with presumably fewer assumptions) such as pointing out an item to change physical copresence from potential or prior to immediate.

While the above assumptions may be sufficient for an expectation of mutual belief and felicitous use of a definite referring expression, they are not sufficient to provide actual mutual belief because of the possibility of error (and possible repair). If **A** makes a reference, she cannot be sure that it will be understood by **B**. Because of this, even if **B** believes he understands the reference (he still might be mistaken) he cannot be sure that **A** believes he does. Each further nested statement introduces more and more uncertainty, and after a while, one of them must be certain to be disbelieved. There is a wealth of linguistic evidence that understandability and attention (or "observability" in Perrault's scheme) are not just mutually assumed. Statements in discourse are often acknowledged by the listener to provide the speaker with evidence that he has been heard and understood. Utterances like "okay", "uh-huh", and "mmh" are often used to acknowledge the previous utterance. With observability assumed, there would be no need to ever make such utterances.

Perner and Garnham [1988] show some additional problems with Clark and Mar-

shall's copresence heuristics. They end up proposing something very much like the shared situation approach from [Lewis, 1969], with the additional restriction that the indications to the population that the situation holds be based on mutual beliefs.

Halpern and Moses [1990] present several notions of group knowledge, ranging from implicit group knowledge to full mutual knowledge. They also offer a proof that mutual knowledge is unachievable in an unsynchronized noisy environment, where communication is not guaranteed. They also investigate weaker notions of common knowledge that are achievable.

2.2.4 Coordinated Activity, Shared Plans and Joint Intention

Although most agree that some sort of collective intentional intentional attitude is useful in formalizing an account of cooperative behavior, it is still fairly controversial what the properties of such an attitude should be and how this collective intention is related to the individual intentions of the participants. Important questions include: how do shared intentions guide individual action, and how can individual beliefs and intentions come together to form shared intentions? This section reviews some previous proposals. Our own attempts at formalizing this notion are presented in Section 7.5, below.

Lewis [1969] defined a *Convention* as a situation in which there is some regularity R in behavior in a population, and everyone conforms to R, everyone expects everyone else to conform to R, and everyone prefers to conform to R, given that everyone else will. A typical example is which side of the road to drive on. In England it is the left side, in America, the right. It doesn't really matter to the drivers which side to drive on, as long as everyone agrees. Coordinated activity is thus seen as individual intention in a state of mutual knowledge about norms. Knowledge of conventions serve to make it in the mutual self-interest of each of the members of the population to follow along.

Grosz and Sidner [1990] take basically the same viewpoint. They extend Pollack's definition of a Simple Plan [Pollack, 1990], to that of a *SharedPlan*, which is formalized as a set of mutual beliefs about the executability of actions and the intentions of particular agents to perform parts of that action. They also present some conversational default rules based on cooperativeness to use communication to add to the shared beliefs.

Searle [1990] starts with the intuition that collective intention is not just a summation of individual intentions. He wants to distinguish between just following a convention and actual cooperative activity. He postulates that *we-intentions* are a primitive form of intentionality, not reducible to individual intentions. There is still a problem of how *we-intentions* can produce the individual intentions necessary for an individual to act.

Cohen and Levesque present their own theory, not in terms of individual intentions (which also aren't primitive in their theory) but in terms of mutual belief and weak mutual goals [Levesque *et al.*, 1990; Cohen and Levesque, 1991b]. Their formulation says that the individuals each have the goals to perform the action until they believe that either it has been accomplished or becomes impossible. Also in the event of it becoming completed or impossible, the agents must strive to make this belief mutual. This framework is also used to explain certain types of communicative behavior such as confirmations as described above in section 2.1.

2.2.5 Obligations

Obligations represent what an agent *should* do, according to some set of norms. The notion of obligation has been studied for many centuries, and its formal aspects are examined using deontic logic. Generally, obligation is defined in terms of a modal operator often called *permissible*. An action is *obligatory* if it is not permissible not to do it. An action is *forbidden* if it is not permissible. An informal semantics of the operator can be given by positing a set of rules of behavior R. An action is obligatory if its occurrence logically follows from R, and forbidden if its non-occurrence logically follows from R. An action that might occur or not occur according to R is neither obligatory nor forbidden. Von Wright, [1951] presented one of the most influential early systems of axioms relating these operators. McCarty [1986] provides an account based on actions rather than states, which also allows the set of obligations to change dynamically, as agents make choices which might incur new obligations. Conte and Castelfranchi [1993] show how obligations share some features of goals and some of beliefs, but cannot be reduced completely to either.

2.3 Conversation Analysis

The primary aim of the subfield of sociology known as *Conversation Analysis*[1] (henceforth CA) has been to study actual conversations and inductively discover recurring patterns found in the data. Although the professed aims seem to be to steer away from intuitions or prior formalization, CA has produced a number of useful insights for how natural language conversation is organized, and which features of conversation a conversant should orient to. Although the conversation analysts do not formulate it in this way, they examine some of the properties of conversation which show it to be the result of interactions among multiple autonomous agents. The rest of this section is devoted to a brief overview of some of the most relevant findings for designing a computational system to converse in natural language.

2.3.1 Turn-taking

Sacks *et al.* [1974] present several observations about the distribution of speakers over time in a conversation. Although there are frequently periods of overlap in which more than one conversant is speaking, these periods are usually brief (accounting for no more than and often considerably less than 5% of the speech stream [Levinson, 1983, p. 296]. Conversation can thus be seen as divided into *turns*, where the conversants alternate at performing the role of speaker. The "floor" can be seen as an economic resource whose control must be divided among the conversants. Although in general conversation (as opposed to more formal communicative settings such as debates, court trials, or classes) there is no predetermined structure for how long a particular turn will last, there are locally organized principles for shifting turns from conversant to conversant.

[1]This gloss of some of the findings of Conversation Analysis comes mainly from [Levinson, 1983]

Turns are built out of *turn constructional units*, which correspond to sentential, clausal, phrasal or lexical syntactic constructions [Sacks *et al.*, 1974, p. 702]. Following a turn constructional unit is a *transition relevance place*, which is an appropriate moment for a change of turn. Subsequent turns can be allocated by one of two methods: either the current speaker can select the next one (as in a question directed to a particular individual), or the next speaker can self-select, as in an interruption or restarting after an unmarked pause.

One important observation about turn-taking is that it is locally managed. The length and structure of a turn is an emergent property of interaction rather than a predetermined structure. The length of one speaker's turn will be determined by the speaker and other conversants who might end things at different times by taking over. A speaker can direct another to speak, but this does not by itself effect a transfer of the turn – the other must pick it up as well. Keeping the stream of talk to mostly be used by a single speaker at a given time is a coordination problem similar to that of two motorists crossing each other's path (though with less drastic consequences for failure). Schegloff [1987] presents an explanation of how conversants re-utter overlapped talk at the beginning of turn transitions.

2.3.2 Adjacency Pairs

Adjacency pairs are pairs of utterances that are [Levinson, 1983, p. 303]:

1. adjacent
2. produced by different speakers
3. ordered into a *first part* and a *second part*
4. typed so that a particular first requires a particular (range of) second(s)

Typical examples of adjacency pairs are question-answer, greeting-greeting, offer-acceptance, and assessment-agreement.

First parts and second parts are connected not by some sort of grammar rule for legal conversations, but because the first will make the second *conditionally relevant.* The following utterance by the speaker after the utterer of the first should be either a second, an explanation that the second is not forthcoming, or something preparatory to a second, e.g., a clarification question. Utterances that come between a first and its second are called *insertion sequences.*

There are two types of seconds that can follow a first. These are known as *preferred* and *dispreferred* responses. Preferred responses are generally direct follow-ups and are unmarked. Dispreferred responses are generally marked with one or more of the following: pauses, prefaces (such as "uhh" or "well"), insertion sequences, apologies, qualifiers (e.g., "I'm not sure but ..."), explanations [Levinson, 1983, p. 334]. Table 2.2 (from [Levinson, 1983, p. 336]) shows some common adjacency pairs with preferred and dispreferred seconds.

First Parts:	Request	Offer/Invite	Assessment	Question	Blame
Second Parts:					
Preferred:	acceptance	acceptance	agreement	expected answer	denial
Dispreferred:	refusal	refusal	disagreement	unexpected answer or non-answer	admission

Table 2.2: Adjacency Pairs

Adjacency pairs can thus serve as contextual resources for interpreting utterances. If a first part has been made, it makes a second conditionally relevant. The next utterance can be checked to see if it forms a plausible second. Markedness or its absence can be seen as pointing to the preferred or dispreferred second.

2.3.3 Repairs

Repairs can be characterized as attempts to fix previous utterances that are perceived to be (possibly) insufficient for conveying what was intended. Repairs include both *clarifications*, in which new information is added, and *corrections*, in which changes are made. Repairs are classified as to who made them (self or other), initiated them (self or other), and how many utterances they are removed from the utterance that they are repairing. In the first (same) turn we can have only self-initiated self-repair. In the second turn, we can have other-repair or other-initiated self-repair. There is also third-turn repair (when the Initiator subsequently determines, in virtue of the other's previous utterance, that he has been misunderstood), and fourth-turn repair, (when the other later realizes that his own interpretation was in error). One can initiate a repair by the other conversant with a *next turn repair initiator* (or NTRI), which seems to be basically the same as a clarification question. Schegloff *et al.* [1977] show that a preference scheme exists for when to perform a repair. The highest preference is to perform self-initiated self-repair in the same turn. The next most preferred is to perform self-initiated self-repair in the transition space between turns. Then other-initiated self-repair in the next turn, by means of an NTRI. The least preferred is other-initiated other-repair.

2.4 Previous attempts to incorporate CA in NLP systems

Although there has been an awareness of the work from Conversation Analysis among some AI researchers for some time (at least since [Hobbs and Evans, 1979]), it is only

recently that several researchers have begun attempting to incorporate the findings from Conversational Analysis into computational systems for understanding natural language or human-computer interaction.

Suchman [1987] contrasts the classical planning framework, characterized by complete knowledge and forming fully specified plans, with the situated nature of real-world action, in which too much is changeable and unknown to plan in complete detail far in advance. In the situated view, "plans are best viewed as a weak resource for what is primarily *ad hoc* activity" [Suchman, 1987, p. ix]. She also presents some of the observations and methods of Conversation Analysis, and uses them to analyze the behavior of a computer program for instructing users of a photocopier, which functions by attributing a state of certain sensors on the machine to a step in one of the possible plans for making different kinds of copies. She finds that many of the problems the users have in understanding the instructions of the system come about as a result of the system not conforming to typical patterns of conversation usage. The system would intend one thing which would be understood as another by the users. Suchman calls for system designers and researchers in conversation planning to use the rules of conversation as resources that the system should orient on.

This call to design interfaces which are based on the observations of conversation analysis has been taken up by several of the researchers whose work appears in [Luff *et al.*, 1990]. Frohlich and Luff [1990] have tried to use the principles of CA in building *The Advice System*, a natural language expert system front-end. Although the system uses mouse-controlled menu-based input, and exercises fairly authoritarian control over what can be said, it at least pays lip service to the findings of CA, including adjacency pairs, turn constructional units, repairs (including NTRIs), standard openings and closings (including pre-closings), and preferred and dispreferred responses. They have "a declarative definition of the interaction between user and system" [Frohlich and Luff, 1990, p. 201] composed of elaborate logical grammar rules which specify legal conversations down to low-level details. These rules serve both to update the context as the conversation progresses and to help the system choose what to do next.

Although the *Advice System* seems to be a step in the right direction, there are several problems with it in practice. First, it is much too restrictive in its input to be called real conversation. Its notion of utterance types is restricted to **Questions**, **Answers**, and **Statements**. Although the designers consider all possible combinations of any of these by speaker and hearer, they reject far too many as being impossible. Though something can only be an answer if there is an outstanding question, and the existence of an outstanding question will tend to make a next utterance be seen as an answer, there seems to be no reason to outlaw statements immediately following questions uttered by the same agent. This is a very common pattern for repairs (e.g. the same speaker successively uttering "Where is the Engine? I mean Engine E3."). The *only* point at which a user can interrupt is at a Transition Relevance Place, whereas in real conversation, that is merely the most common and expected place. The menu-based input also trivializes the interpretation problem, and it's unclear why the user should ever have to make a repair. Empirical testing will show if users find the *Advice System* usable or not, but it may well suffer from the same problems that Suchman's copier

system suffers from: using familiar patterns in unfamiliar ways, ending up misleading the user.

Raudaskoski [1990] describes an attempt to study local repair mechanisms in a telephone message-leaving system. She allowed five different kinds of repair initiators, and ran a simulated experiment where a person acted as intermediary, typing input which was spoken by the user, and reading the responses of the system over the phone. The experiment didn't work very well, mainly because the system could only interpret a very small set of inputs, which the users generally went beyond. Also, the repair mechanisms didn't work very well: the full variety was not used, and those that were used often led to misunderstanding. One of the system's repair initiators seemed too much like a confirmation, so the user thought she had succeeded in leaving the message and went on to the next message but the system was still hoping to get the user to start over.

Cawsey [1990] describes the *EDGE* system, which is an expert/advice giver, that combines AI planning with CA local interactions. It has a model of discourse structure based on that of Grosz and Sidner [1986], and plan schemas, which it uses to construct explanations. It also allows local interactions, including forcing the user to mouse-click acknowledgement after every utterance, and allowing the user to break in with repair initiators. Planning is done when required (e.g. to fill in the content for a user requested repair), not in advance.

Cawsey [1991] uses the endorsement based ATMS model for belief revision presented by a belief revision scheme presented in [Galliers, 1990] to model third and fourth turn repair. The belief revision scheme keeps a set of endorsements with each assumption and, when conflict occurs, throws out the assumption set with the weakest endorsement. Cawsey uses a Speech act plan-recognition system based on that of [Perrault and Allen, 1980], but makes the interpretations *assumptions*, subject to change if conflict occurs. Thus the system can change its interpretations of prior utterances to bring them in line with new evidence.

Probably the most ambitious synthesis of ideas from both Conversation Analysis and AI theories of speech acts and mental states is that of McRoy [1993]. Each speech act is paired with both a set of *lintentions*, representing the mental attitudes expressed by the speaker in performing the speech act, and a set of *lexpectations*, representing the acts which are expected to follow the speech act in the course of a dialogue. These expectations come from the adjacency structure pairs from CA. A uniform abductive framework is used to model the discourse strucutre revision and to be able to generate appropriate speech acts, interpret surface speech act utterances using discourse coherence, and detect and repair misunderstandings, including third and fourth-turn repair.

2.5 Grounding in Conversation and the Contribution Model

Clark and several of his colleagues have looked at coordination and collaborative activity in conversation, making explicit reference to both the traditions of Conversation Anal-

ysis and Speech Act Theory [Clark and Wilkes-Gibbs, 1986; Clark and Schaefer, 1989; Brennan, 1990; Clark and Brennan, 1990; Clark, 1992]. They try to identify several principles serving to guide collaborative behavior to account for the kinds of things observed by the Conversation Analysts.

One of the points that they make is that conversants need to bring a certain amount of common ground to a conversation, in order to understand each other. They call the process of adding to this common ground *grounding*. Grounding can be seen as adding to the mutual beliefs of the conversants (in fact it is glossed this way in [Clark and Schaefer, 1989]), but it seems reasonable to make a distinction. Though mutual belief, as defined by any of the proposals described in Section 2.2.3, is probably sufficient for common ground, it may be that only some weaker notion is actually necessary, and that we can have some sort of common ground without full mutual belief.

Clark and Schaefer [1989] present a model for representing grounding in conversation using *contributions*. Contributions are composed of two parts. First, the contributor specifies the content of his contribution and the partners try to register that content. Second, the contributor and partners try to reach the *grounding criterion*, which Clark and Schaefer state as follows, "The contributor and the partners mutually believe that the partners have understood what the contributor meant to a criterion sufficient for the current purpose" [Clark and Schaefer, 1989, p. 262]. Clark and Schaefer divide the contribution into two phases as follows (for two participants, **A** and **B**) [Clark and Schaefer, 1989, p. 265]:

Presentation Phase: **A** presents utterance **u** for **B** to consider. He does so on the assumption that, if **B** gives evidence **e** or stronger, he can believe that **B** understands what **A** means by **u**.

Acceptance Phase: **B** accepts utterance **u** by giving evidence $\mathbf{e}'$ that he believes he understands what **A** means by **u**. He does so on the assumption that, once **A** registers evidence $\mathbf{e}'$, he will also believe that **B** understands.

Clark and Schaefer claim that once both phases have been completed, it will be common ground between **A** and **B** that **B** understands what **A** meant. Each element of the contribution may take multiple conversational turns and may include whole embedded contributions. Rather than a straightforward acceptance, **B** can instead pursue a repair of **A**'s presentation, or ignore it altogether. **B**'s next turn, whether it be an acceptance, or some other kind of utterance, is itself the presentation phase of another contribution. Thus **A** must accept **B**'s acceptance, and so on.

There are different types of evidence which can be given to show understanding. The main types considered by Clark and Schaefer are shown in Table 2.3, in order from strongest to weakest.

The strength of evidence needed for grounding depends on several factors, including the complexity of the presentation, how important recognition is, and how close the interpretation has to be. They try to avoid infinite recursion in accepting acceptances by invoking the following **Strength of Evidence Principle:** The participants expect that, if evidence $\mathbf{e_0}$is needed for accepting presentation $\mathbf{u_0}$, and $\mathbf{e_1}$ for accepting presentation of $\mathbf{e_0}$, then $\mathbf{e_1}$ will be weaker than $\mathbf{e_0}$.

1	**Display**	B displays verbatim all or part of A's presentation.
2	**Demonstration**	B demonstrates all or part of what he has understood A to mean.
3	**Acknowledgement**	B nods or says "uh huh", "yeah", or the like.
4	**Initiation of relevant next contribution**	B starts in on the next contribution that would be relevant at a level as high as the current one.
5	**Continued attention**	B shows that he is continuing to attend and therefore remains satisfied with A's presentation.

Table 2.3: [Clark and Schaefer, 1989, p. 267]: Types of Evidence of Understanding

Clark and Wilkes-Gibbs [1986] present a **Principle of Least Collaborative Effort** which states that "In conversation the participants try to minimize their collaborative effort – the work that both do from the initiation of each contribution to its mutual acceptance." This principle is contrasted with Grice's maxims of quantity and manner, which concern themselves more with least effort for the speaker.

Clark and Brennan [1990] show how the principle of least collaborative effort can help in explicating the preferences for self-repair shown by [Schegloff *et al.*, 1977] (described above in Section 2.3). They also show how this principle predicts different types of grounding mechanisms for different conversational media, depending on the resources that are available and their costs in those different media.

Brennan [1990] provides experimental evidence for how grounding takes place in conversational tasks, and the principles described above. She has a computer-based location task, where one party (called the *director*) must describe where on a map the other (the *matcher*) is to point his cursor. The experiment is broken down along two dimensions: familiar vs. unfamiliar maps, to change the grounding criterion, and trials where the director can see where the matcher is vs. trials where the director cannot, and must rely on verbal descriptions from the matcher, to change the strength and type of evidence available for accepting presentations. As might be expected, participants took longer to describe and find locations on the unfamiliar map, and the grounding process was shorter where more direct evidence was available.

2.6 Deficiencies of the Contribution Model

Although the contribution model is perhaps the first explicit model of how grounding takes place and why acknowledgements occur, it still is lacking in a number of particulars. For one thing, it is often hard to tell whether a particular utterance is part of the presentation phase or the acceptance phase. Self-initiated self-repair and other-agent completions are considered part of the presentation phase, but other-repair and other-initiated self-repair are part of the acceptance phase. Either one can have embedded contributions, in the form of insertion sequences or clarification sub-dialogues, so, in the case of an other-initiated self-repair, it's hard to tell whether it is part of the presentation phase or the acceptance phase. We often need to look at large segments of the conversation, both before and afterwards before deciding how a particular utterance fits in.

Since Clark and Scahefer assert that each signal (including acceptances) must itself be a presentation which needs acceptance, it is not clear that contributions are *ever* really complete. For example, in the simplest case, *contributions by turns*, Speaker A's first turn is a presentation part of a first contribution. Speaker B's following turn is an acceptance part of that contribution, but also is the presentation part of a next contribution. What is unclear is whether this second utterance must be accepted in order to fulfill its acceptance function. If so, as Clark and Shaefer seem to imply, then not even the next utterance by A will completely ground the first contribution: this acceptance of the acceptance will itself need to be accepted, and so on, ad infinitum. If it is possible, as they suggest, that some acceptances need not be accepted themselves, then this opens the possibility that the acceptance part of an utterance need not be itself accepted (though any actual intended next contribution which is part of the same utterance would still have to be accepted). Some utterances which are merely acceptances might not be presentations as well.

The model also seems insufficient to use as a guide for an agent in a conversation deciding what to do next based on what has happened before. Realizing that a presentation has been made but has not yet been accepted can lead one to initiate the acceptance phase, but it's not clear when a presentation or acceptance is complete, or whether the knowledge of being in the presentation phase or acceptance phase has any consequences for what should be uttered.

In the next chapter, we will attempt to solve these problems using a modified model of grounding. We replace the *presentation* and acceptance phases with more atomic utterance level acts that build up complete contributions according to a grammar which will show, at every stage in the process, whether the contribution is grounded, what acts are allowed to follow, and what it will take to become grounded.

3 Towards a Computational Theory of Grounding

This chapter presents the foundations of a computational theory of grounding that is based on how individual utterance level actions contribute towards the grounding process. The central theoretical constructs introduced are the **discourse unit** (DU), which is the level of structure at which conversational content is grounded, and **grounding acts**, which are the actions performed in producing particular utterances which contribute to this groundedness. This chapter presents a theory of how DUs are constructed from a sequence of grounding acts, providing constraints on possible grounding act sequences, a decision procedure for determining whether a discourse unit is grounded, and setting up expectations for preferred sequences which will lead to a grounded unit from a particular ungrounded but accessible discourse unit. It thus forms a theory of grounding which is competitive with that of Clark and Shaefer described in Section 2.5.

Section 3.1 builds up a recursive transition network model of discourse unit construction, starting from a model like Clark and Shaefer's contribution model and amending it to handle single utterance grounding acts. Section 3.2 presents a finite-state model which is roughly equivalent to the recursive model but computationally simpler. It then continues by describing how this model can be used as a protocol for grounding. Section 3.3 describes a cognitive model of grounding that is based on the mental states and processes of agents involved in conversation. The grounding acts in the previous section are related to their effects on the cognitive model and the reasons why an agent might decide to produce them to accomplish conversational purposes. Section 3.4 shows how this model can be used to explain the distribution of utterance acts described in Section 3.2. Section 3.5 presents some problematic conversation sequences for this account of grounding, considers some alternative accounts and suggests some simple solutions.

3.1 Tracking Grounding with Discourse Units

From a processing point of view, the main deficiency of Clark and Schaefer's **contribution** model (described in Section 2.5) of grounding is that there is no easy way to tell the "state" of the current contribution while engaged in a conversation. Although we might represent a contribution as a transition network such as that in Figure 3.1, with a grounded contribution being one in the final state, F, this is not sufficient to monitor on-line conversation.

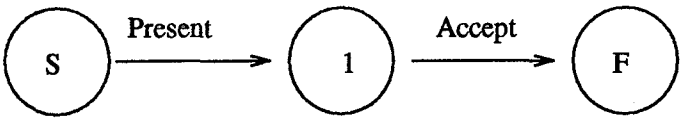

Figure 3.1: Transition Network for Contributions

We know that to start a contribution, a **presentation** must be performed primarily by one agent, whom we will call the *Initiator* (I) and then an **acceptance** must be performed primarily by the other agent, whom we will call the *Responder* (R), but what is less obvious is how to tell when a presentation has been performed. Another way of looking at this question is: given an utterance by the initiator, how does it function as part of the current contribution? Does it start, continue, or complete a presentation? Unfortunately, there is no way to recognize whether a presentation is complete, just by looking at an utterance itself. Consider the following sequences in examples (6) and (7).

(6) I: Move the boxcar to Corning

I: and load it with oranges

R: ok

(7) I: Move the boxcar to Corning

R: ok

I: and load it with oranges

R: ok

Since, according to Clark and Shaefer, the sequence in example (6) is a single contribution, that means that the presentation phase must encompass both of the utterances by the initiator. However, in example (7), there are two contributions with two separate presentations by I[1], and thus the first utterance by the initiator is a complete presentation. Since these sequences are identical up to the second utterance, there is, in general, no way to tell whether a presentation is complete until another action starts. This becomes a more serious matter because a contribution must be composed of actions by both participants, and thus there must be some way for the individual participants to determine an appropriate next action, given the current state.

Since the notion of a *presentation* does not seem to provide much help in processing contributions, we will replace *contributions* with a similar notion: **discourse units**. Like contributions, discourse units (DUs) also represent the unit of conversation at which grounding takes place, but they are composed of individual utterance-level actions

[1]Clark & Shaefer would also analyze the acknowledgements by R as the presentation phases of separate contributions.

rather than (potentially recursive) phases. The opening utterance of a DU will be called an `initiate` act, subsequent utterances which add on new material in a presentation (such as the second utterance in (6)) will be called `continue` acts. We will also call the act of claiming understanding of a previous utterance an `acknowledgement` (or `ack` act), regardless of the type of utterance it is and whatever other functions it might have in other discourse units. Thus a first approximation at a transition network for DUs (ignoring repairs) would look like figure 3.2, where the actor is given in parentheses (I for initiator, R for responder).

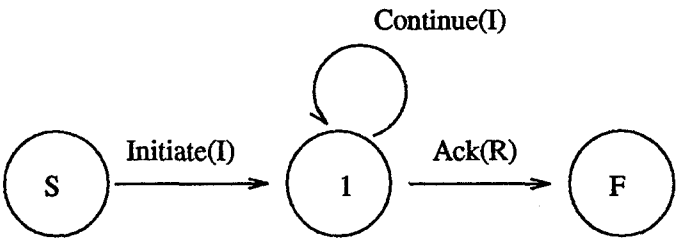

Figure 3.2: Initial Transition Network for DUs

This network would be sufficient to determine the grounding status of any sequence of connected utterances followed by their acknowledgement. From this basic model, we can now add in a facility to allow repairs by both parties. First, we will want to add a "dead" state, to cover cases in which the initiator retracts the proposed contribution or the conversation proceeds in such a way that the material can no longer be grounded (but would have to be brought up again in a new contribution). We call an utterance that makes the material in the DU ungroundable a `cancel` act.

It will also be a simple matter to incorporate self-initiated self-repair to this network. These repairs will change the material under consideration (whereas a `continue` would just add additional material), but, still, the responder can acknowledge the updated contents with a single acknowledgement. Figure 3.3 gives an amended network which allows cancels and self-initiated self-repairs.[2]

To allow other-initiated self-repair and other repair, we need a slightly more complex theory. Unlike the simpler cases above in which, once a DU has been initiated, it could always be grounded with a single acknowledgement by the responder, these cases will require other utterances before being grounded. The question thus arises as to how much extra complexity is needed. Will adding an extra few states to the network suffice, or is a recursive network (or something more) required?

For presentational simplicity, we will start with a recursive transition network, following Clark and Schaefer's model of nested contributions. A repair by the responder must be acknowledged by the initiator in order to be grounded. Thus, we add a sub-

[2]It remains open to question whether the link from the final state, F, to the dead state, D, in Figure 3.3 and subsequently should be allowed. This is equivalent to the question of whether an initiator can cancel a proposed contribution after it has been acknowledged, or whether the effect of changing assumed mutual belief can result only from the grounding of new DUs. Further study of these sequences is still required to answer the question conclusively.

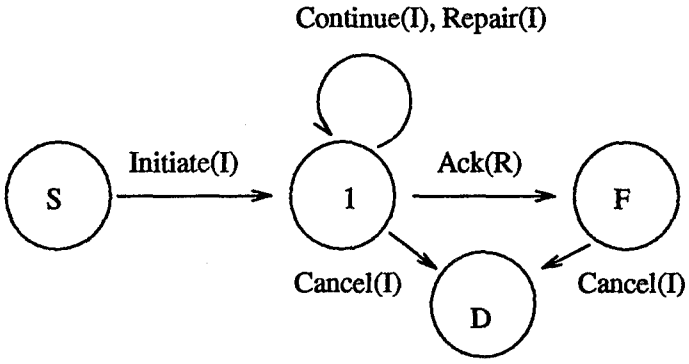

Figure 3.3: Transition Network for DUs with self-initiated self-repair and cancels

network that looks identical to our original DU network, except that the initial act is a `repair` instead of an `initiate` (one can think of this `repair` as an `initiate` of a *repair* insertion sequence). In addition, this sub-DU will have the initiator and responder switched, since it is the responder of the main unit who initiates this repair. Another recursive network needed is one for other-initiated self-repair. In the case of a repair request, rather than acknowledging, the initiator will have to perform a repair before the responder can acknowledge the whole DU.

We represent recursive transitions (as opposed to single utterance transitions) in all caps. In addition, the agent who plays the role of initiator of the sub-network is shown within square brackets. Thus REPAIR[R] indicates a transition in which the REPAIR network must be followed and in which the responder of the current network is the initiator of the REPAIR network. Figure 3.4 shows the modified DU network, Figure 3.5 shows the network for (other) repairs, and Figure 3.6 shows the network for repair requests (other-initiated self-repairs), abbreviated as *REQ-REPAIR*.

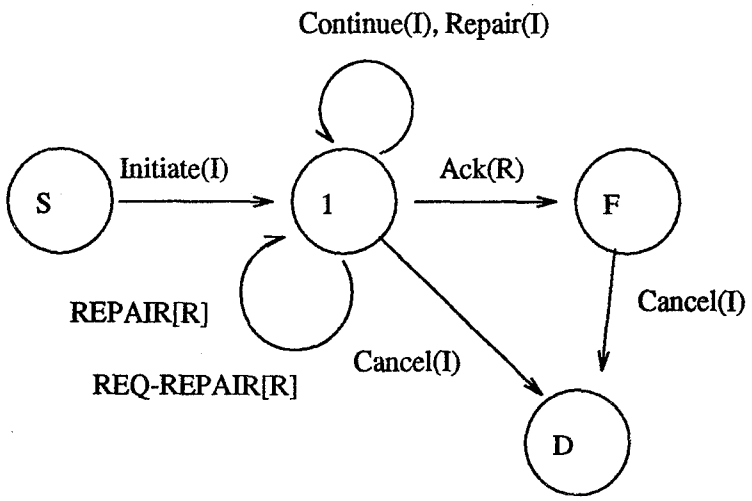

Figure 3.4: Transition Network for DUs with recursive repairs

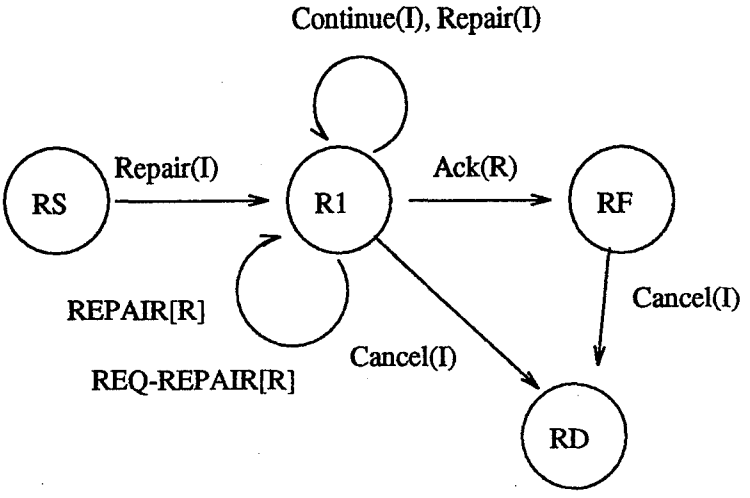

Figure 3.5: Transition Network for REPAIR Sub-DU

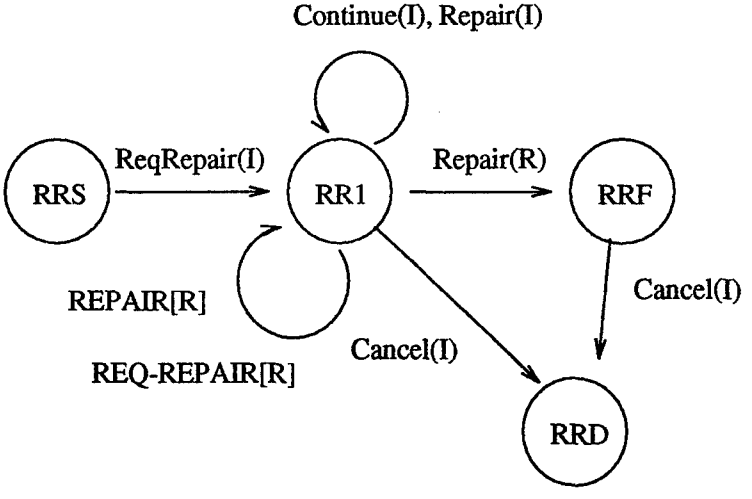

Figure 3.6: Transition Network for REQ-REPAIR Sub-DU

Following the REPAIR[R] means traversing the REPAIR sub-DU network, from the start state (RS) to the final state (RF). This complete traversal will cause the sub-DU network to be *popped* and the parent network will now be in the resulting state. Also, if the repair is cancelled (ending up in state RD), then it is as if the repair network were never started, the sub-DU network is popped, and the parent network is in the state it was in when things started. Note that these extra networks are recursive since the REPAIR and REQ-REPAIR networks also allow repairs and repair requests.

We will also add two more capabilities to the main network. First, we'd like to allow requests for acknowledgement by the initiator. These will signal that the speaker believes the current unit is complete and wants an acknowledgement before proceeding. Also, we will allow further acts from the final state. After an acknowledgement, it is usually possible to add extra acknowledgements, while remaining grounded. It is also possible for a short while to *reopen* a completed DU with a further repair. We thus amend Figure 3.4 to Figure 3.7. This ability helps to alleviate one of the problems with Clark and Shaefer's *Contribution* model. Since a contribution was not deemed complete until nothing more was added to it, the acceptance phase was not considered complete after an acknowledgement, because the initiator could still acknowledge in return or reject the acknowledgement with a repair. However, these actions are not mandatory, so we represent the state of the DU after each act, and can determine whether a DU is considered grounded by tracking the sequence of acts that have occurred as part of the DU.

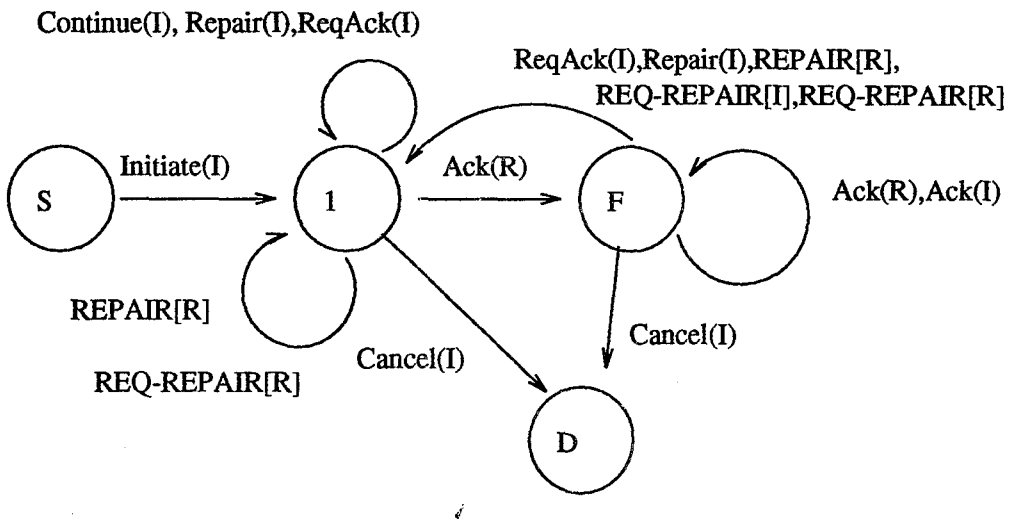

Figure 3.7: Modified Transition Network for DUs with recursive repairs

3.2 A Finite-State Theory of Grounding

An important question is whether the recursion is really necessary. Is this unbounded depth of nesting of repairs actually used in dialogue, or do participants always get lost after a few levels and have to cancel and start again? If so, then a finite-state model

might be more efficient. A more serious question is whether such nesting really occurs at all, or whether everything is at the top level: does one really have to *pop* out of each level of repair before proceeding with the previous repair? In this section, we develop a finite-state model which is competitive with the recursive model concerning coverage of the observed sequences in the TRAINS corpus.

For a finite-state model, we do need at least some extra states beyond those of Figure 3.7 to represent the results of repair. We can begin by folding in the repair networks in Figures 3.5 and 3.6. After a `reqRepair` by the responder in the main DU (the initiator of the REQ-REPAIR sub-DU), we need a `repair` by the initiator (the responder of the sub-DU) to get back to state 1 in the main DU. We can achieve this by adding a state 2 to the network in Figure 3.7, which is reached by a `reqRepair` by the responder and from which a `repair` by the initiator will return the DU to state 1. This state 2 thus corresponds to state RR1 in the first level REQ-REPAIR network. In addition, we will need another state for the result of a `repair` by the responder, corresponding to state R1. We will call this state 3. The resulting network will look something like Figure 3.8, although, for readability, this only shows the transitions to and from state 1.

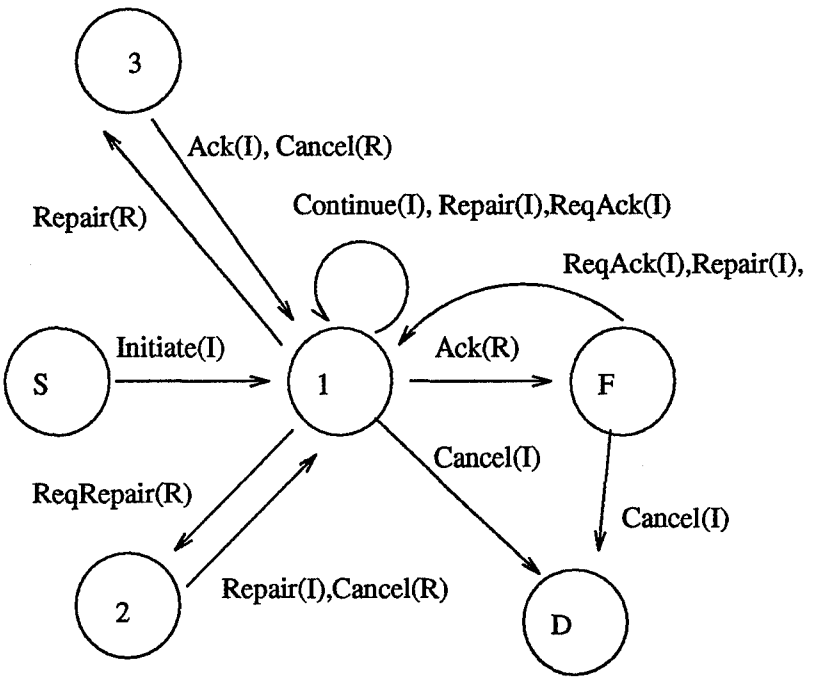

Figure 3.8: Finite-State Network with single level of repairs

This network is identical to the recursive network for single depths of nesting. However, we can extend the finite-state model to cover additional depths of nesting. First, we will also want to allow the possibility of a repair request by the initiator of the DU after some sort of response by the responder. This would correspond to a REQ-REPAIR[I] network from state F, or a REQ-REPAIR[R] network from the first level of either the REQ-REPAIR[R] (state 2) or REPAIR[R] (state 3) networks. This will be

state 4, an analogue of state 2, in which a repair by the responder is expected.

Finally, we will need to decide what should happen when further repairs are performed. A plausible option is to treat all repairs as cumulative, operating on the content on the *floor*, and merely (perhaps) shifting the initiative and discourse obligations. State 1 represents initiative with the initiator and no discourse obligations (beyond those for the responder to acknowledge); state 2 represents initiative with the initiator and a discourse obligation for the initiator to repair; state 3 represents initiative with the responder and and no discourse obligations (beyond those for the initiator to acknowledge); state 4 represents initiative with the responder and a discourse obligation for the responder to repair.

So, to Figure 3.8, we add a link from state 3 for a repair by the initiator to state 1. In addition, in state 3, we consider the DU grounded if the initiator acknowledges the repair, without requiring an additional acknowledgement by the responder. Thus we change the arc for Ack(I) to go directly from state 3 to state F instead of state 1. Figure 3.9 shows the network from the point of view of state 1, Figure 3.10 from that of state 2, Figure 3.11 from that of state 3, Figure 3.12 from that of state 4, and Figure 3.13 from that of state F. Table 3.1 shows a transition table for the entire network.

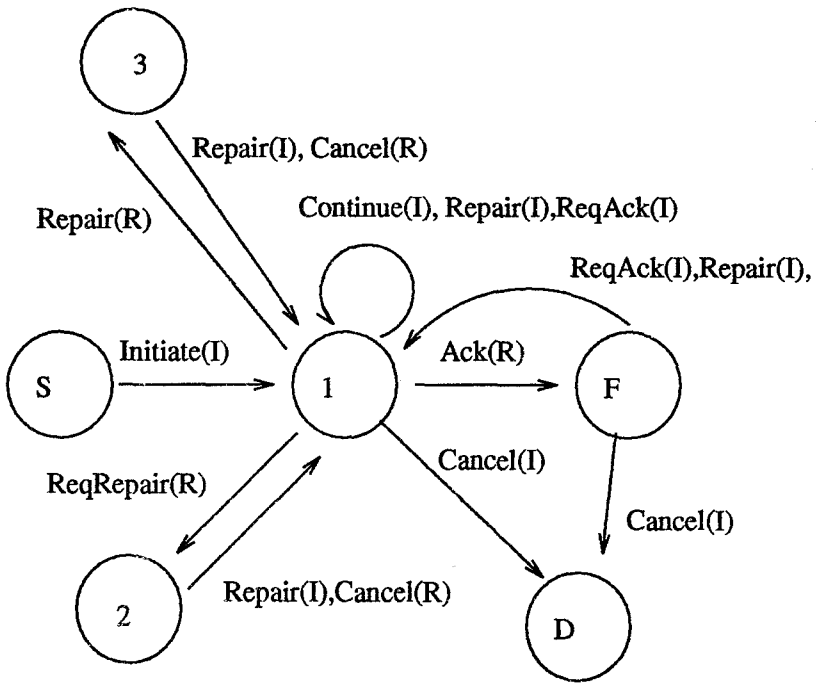

Figure 3.9: Transition Network to and from State 1

Agents may take different roles in different DUs in a mixed-initiative conversation. A *completed* discourse unit is one in which the intent of the initiator becomes mutually understood (or *grounded*) by the conversants. An *open* DU is one which has been initiated but to which agents may still contribute.

While there may be some confusion among the parties as to what role a particular

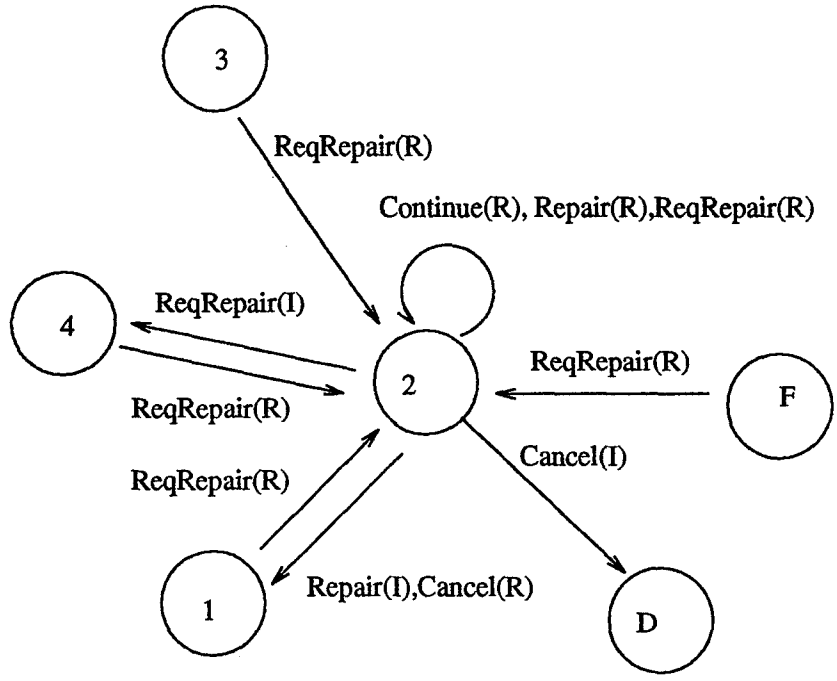

Figure 3.10: Transition Network to and from State 2

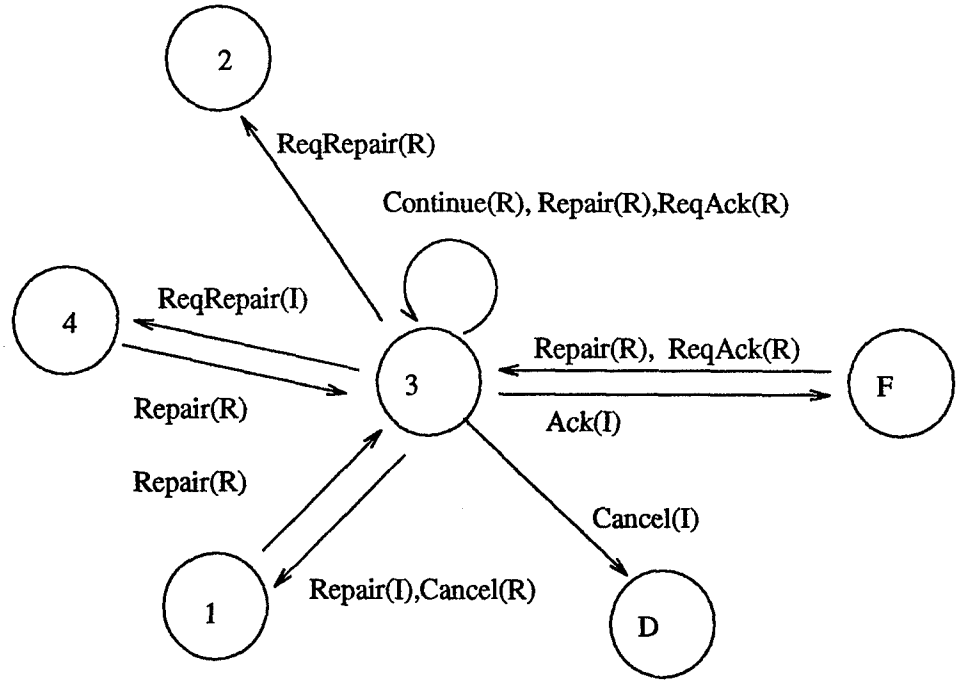

Figure 3.11: Transition Network to and from State 3

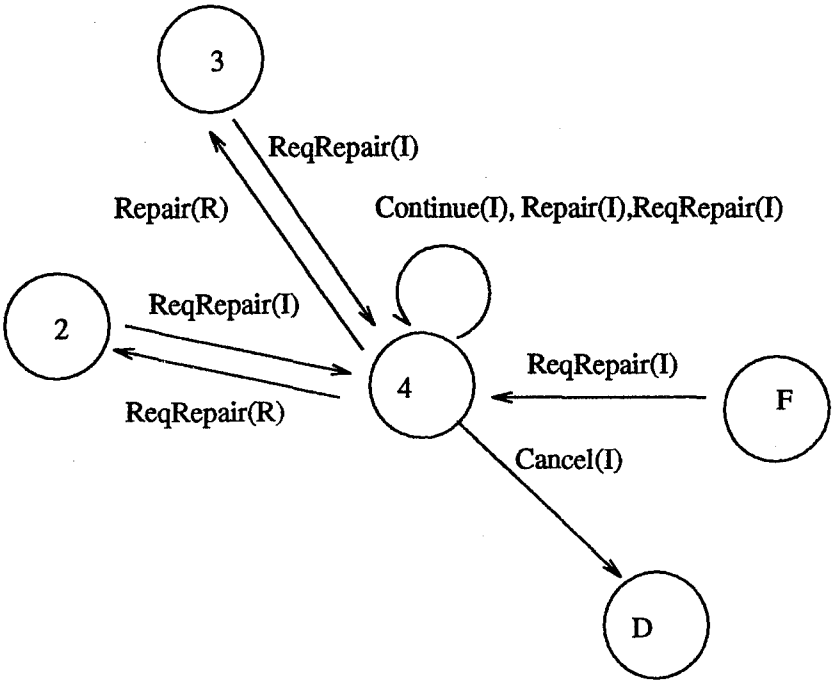

Figure 3.12: Transition Network to and from State 4

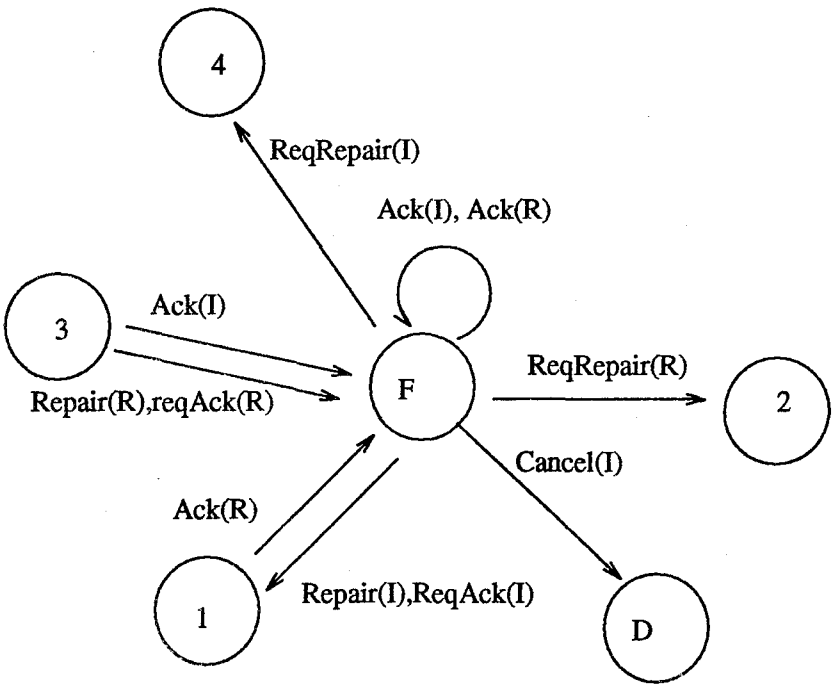

Figure 3.13: Transition Network to and from State F

Next Act	In State						
	S	1	2	3	4	F	D
Initiate^{I}	1						
Continue^{I}		1			4		
Continue^{R}			2	3			
Repair^{I}		1	1	1	4	1	
Repair^{R}		3	2	3	3	3	
ReqRepair^{I}			4	4	4	4	
ReqRepair^{R}		2	2	2	2	2	
Ack^{I}				F	1*	F	
Ack^{R}		F	F*			F	
ReqAck^{I}		1				1	
ReqAck^{R}				3		3	
Cancel^{I}		D	D	D	D	D	
Cancel^{R}			1	1		D	

*repair request is ignored

Table 3.1: DU Transition Diagram

utterance plays in a unit, whether a discourse unit has been completed, or just what it would take to complete one, only certain patterns of actions are allowed, as shown in Table 3.1. Impossible actions are represented in the table by blanks. For instance, a speaker cannot acknowledge his own immediately prior utterance. He may utter something (e.g. "ok") which is often used to convey an acknowledgement, but this cannot be seen as an acknowledgement in this case. Often it will be seen as a request for acknowledgement by the other party. Similarly, a speaker cannot `continue` an utterance begun by another agent. The speaker could produce an utterance which contains a syntactic continuation of, or conceptually related material to another utterance by another agent, but this would not be a `continue` act. Depending on context, it would be interpreted as either an acknowledgement (*e.g.*, if one is just completing the other's thought), a `repair` (if one is correcting to what *should* have been said), or an `initiate` of a new DU (if it provides new information).

If one is in a state and recognizes an impossible action by the other agent, there are two possibilities: the action interpretation is incorrect, or the other agent does not believe that the current DU is in the same state (through either not processing a previous utterance or interpreting its action type differently). Either way, this is a cue that repair is needed and should be undertaken. One also always has the option of initiating a new DU, and it may be the case that more than one DU is open at a time. If a DU is left in one of the non-final states, then its contents should not be seen as grounded.

As shown above, we can identify at least seven different possible states for a DU to be in. These states can be distinguished by their relevant context: what acts have been

State	Entering Act	Preferred Exiting Act
S	———	$\text{Initiate}^{\text{I}}$
1	$\text{Initiate}^{\text{I}}$	Ack^{R}
2	$\text{ReqRepair}^{\text{R}}$	Repair^{I}
3	Repair^{R}	Ack^{I}
4	$\text{ReqRepair}^{\text{I}}$	Repair^{R}
F	$\text{Ack}^{\{\text{I,R}\}}$	$\text{Initiate}^{\{\text{I,R}\}}$ (next DU)
D	$\text{Cancel}^{\{\text{I,R}\}}$	$\text{Initiate}^{\{\text{I,R}\}}$ (next DU)

Table 3.2: Summary of Discourse Unit States

performed and what is preferred to follow, as shown in Table 3.2. State S represents a DU that has not been initiated yet. State F represents one that has been grounded, though we can often add more to this unit, as in a further acknowledgement or some sort of repair. State D represents an abandoned DU, ungrounded and ungroundable. The other states represent DUs which still need one or more utterance acts to be grounded. State 1 represents the state in which all that is needed is an acknowledgement by the responder. This is also the state that results immediately after an initiation. However, the responder may also request a repair, in which case we need a repair by the initiator before the responder acknowledges; this is state 2. The responder may also repair directly (state 3), in which case the initiator needs to acknowledge this repair. Similarly the initiator may have problems with the responder's utterance, and may request that the responder repair; this is state 4.

It is important to note that these states are states *only* of the DU itself. We are not claiming that aspects of discourse other than grounding could be modeled with a finite-state automaton. Also, these states do not refer to the "state" of a computational agent engaged in conversation. Any computational agent (e.g., that proposed in Chapter 5) will have to pay attention to more than just grounding, and even a grounding module will need to pay attention to multiple DUs, each of which will have an associated state.

This finite-state machine has been constructed by analyzing common sequences of utterances in the TRAINS corpus, guided by intuitions about possible continuers and what the current state of knowledge is. It serves as an alternative structural account of grounding to Clark and Schaefer's contribution model. This network serves mainly as guide for interpretation, though it can also be an aid in utterance planning. It can be seen as part of discourse segmentation structure maintained as part of the context of a conversation. It can be a guide to recognizing which acts are possible or of highest probability, given the context of which state the conversation is currently in. It can also be a guide to production, channeling the possible next acts, and determining what more is needed to see things as grounded. It is still mainly a descriptive model; it says nothing about when a repair should be uttered[3], only what the state of the conversation is when one is uttered. We can evaluate the correctness of this model by checking to see how it

[3]Except in the obvious case after a ReqRepair (states 2 and 4).

would divide up a conversation, and whether it seems to handle acknowledgements and repairs correctly. We can also evaluate its utility for processing: whether it serves as a useful guide or not. The type of behavior it describes can also be analyzed in terms of the preconditions and effects of actions, as sketched in Section 3.3, but having an explicit model of the nature given here may serve to help repair interactions, and make processing more efficient.

3.3 A Cognitive Model of Grounding Act Processing

In order to model the grounding process, including the planning, production, and recognition of the grounding acts from the previous section, we must use several of the mentalistic notions described in Section 2.2, including *beliefs* and *mutual beliefs*, *intentions*, and *obligations*. In addition, we will assume the existence of several mental processes that can manipulate these attitudes as the agents engage in conversation. We will require an *utterance planner* to produce from communicative intentions, natural language utterances that can help realize those intentions in the current context. Once the utterance has been planned, we need a *language producer* to actually perform the utterances. Finally, we will need a *language interpreter* which can observe linguistic behavior and decide what actions were performed.

Conversation will generally start from *intentions* of an agent to have some content added to the common ground (*mutually believed*). Then the utterance planner will decide on utterances and the language producer will act in accordance with the decisions. The language interpreter will monitor the produced utterances to see if the actions are in accord with the intentions: interpreting the utterances by both the same agent and the other agent will also provide new information about the beliefs and intentions of the agents. Utterances may also introduce obligations, including *discourse obligations*, which involve further participation in the conversation.

Figure 3.14 gives a schematic of the types of cognitive states necessary for tracking grounding using the actions introduced above in Section 3.1. This figure represents the grounding process from the point of view of an agent X. The other agent that X is communicating with is Y. The boxes represent distinct mental attitudes, which are related by different inference processes. Performance of grounding acts can affect the contents of these attitudes, including the transfer information from one to another. Box 6 represents (X's beliefs about) mutual beliefs of X and Y. Box 1 represents information that X intends be mutually believed. Box 3 represents (X's beliefs about) Y's beliefs about what X intends be mutually believed. Box 2 represents (X's interpretations of) the contents of utterances that X makes in order to change Y's beliefs and thereby bring about mutual beliefs. Box 4 represents (X's interpretations of) the contents of Y's utterances, and Box 5, (X's beliefs about) Y's intentions. Also shown are the discourse obligations of the agents. The arrows represent the main information flow that results from the performance of grounding acts.

The grounding process is started when one party or the other makes an utterance which initiates a discourse unit. X will decide to initiate a DU if there is something in

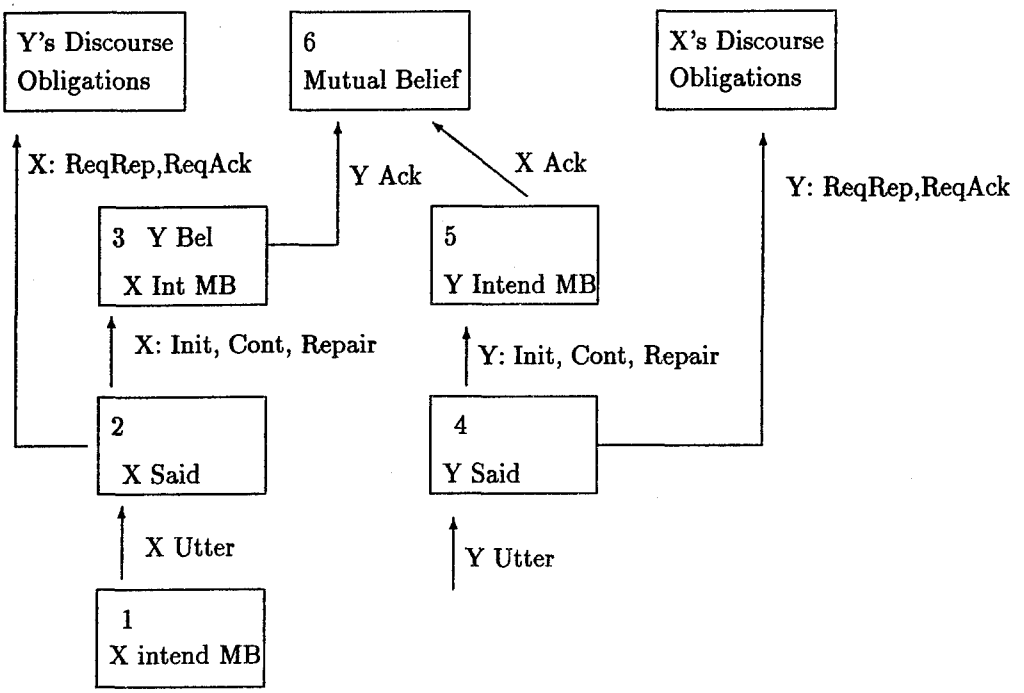

Figure 3.14: Architecture of X's Model of Conversation with Y

Box 1 which is not (either partially, other than Box 6, or completely) part of any of the other attitudes represented in this diagram, and the proper contextual factors apply (X has or can take the turn and there are no outstanding discourse obligations or other goals which are more immediate). X will invoke the utterance planner to come up with an utterance or sequence of utterances that will convey X's intention to Y. When X actually performs the (first) utterance, we will have in Box 2 the interpretation of that utterance. This will most likely be the same as was intended (if the utterance planner is good), but might not be, due to resource constraints on the utterance planning and language production processes. If, somehow, the interpretation of the utterance is different from what was intended, that will provide the basis for planning some sort of repair that can clarify or correct the previous utterance. In addition, it might prove desirable to split up the conveyance of content into several utterances, in which case the content of Box 2 will be only a part of the intended content in Box 1.

Starting with the interpretation of X's own utterance in Box 2, if the interpretation is a "content" act, such as `initiate`, `continue`, or `repair`, then the next step is to do plan recognition and inference using (X's beliefs about) Y's beliefs, to see what Y is likely to believe about X's intentions[4]. The result of this plan recognition, including the act

[4]It might seem that this step is unnecessary, and that the utterance planner should have included this reasoning to produce the utterance in the first place. While, it is true, an ideal utterance planner would always calculate the precise effect of the utterance on the intended audience, and a perfect language production system would always produce exactly the planned utterance, our model will also cover the real-world case in which adequate planning is not always performed and execution errors are possible. Some of the stumbling blocks to perfect planning include complexity and time criticality, incomplete

interpretation and its implicatures, will be placed in Box 3. Now if Box 3 contains the same content as Box 1, X believes that her communication was adequate, and must wait for (or prompt with a `reqAck`) an acknowledgement from Y that Y correctly understood. If, on the other hand, the contents of Box 3 are not the same as those of Box 1, that is further motivation to make a subsequent utterance, which might involve more utterance planning. The subsequent utterance may come out as a `repair`, a `continue`, or even a new `initiate`, depending on the particular differences between actual intentions and the perceived intentions attributed to the other agent. These subsequent utterances would also be subject to the same processes of interpretation and inference leading back through Box 2 to Box 3.

When Y produces an Utterance, X's interpretation of this utterance goes into Box 4. If there is some problem with the interpretation process, such as no plausible interpretation, or no evidence to choose between two or more possible interpretations, this will provide the basis for the utterance planner to come up with some sort of repair initiator, most probably a `reqRepair`, but perhaps (if contextual factors indicate what Y *should have* said) a `repair`.

Once X thinks that she understands Y's utterance, what happens next depends on the actions that X thinks Y has performed. If Y has performed an `ack`, then the appropriate contents of Box 3 (i.e., the contents that Y acknowledged) are moved to Box 6 (MB). If the utterance is an `initiate`, `continue`, or `repair`, then X will do plan recognition and inference and put the results in Box 5. X can make the contents of Box 5 grounded by uttering an acknowledgement, moving these contents on to Box 6. If Y's utterance is either a request for acknowledgement or a request for repair, this will give evidence for more inference to be performed on the contents of Boxes 3 and 5 (taking into account new inferences about Y's beliefs arising from the observation of Y's utterance), as well as adding *discourse obligations* for X to perform or respond to the requested action.

Table 3.3 summarizes the reasons for X to perform particular grounding acts, and shows the effects of performance. For each of X's actions, after coming up with the intention to perform the action, X will first plan the utterance and then perform it, then interpret the actual utterance and place the results in Box 2. Further effects depend on the type of action, and are described in the third column.

Table 3.4 shows the effects of Y's actions. For all of these actions, the interpretation of the utterance will start in Box 4. Acts by Y may also provide additional evidence about Y's mental state, and thus present an opportunity for X to revise her beliefs about the contents of Boxes 3 and 5 from previous utterances. This may lead to an incentive for X to produce a third-turn repair (if the changes are to Box 3) or fourth-turn repair (if the changes are to Box 5).

---

information, and simultaneous action. It might be that more information is available to the reasoner after the utterance is produced than is present at the time the utterance is planned.

Action	Reason	Effects
`Initiate`	Item in [1], no part of item in [3] or [5]	Put contents of act in [3]
`Continue`	Item in [1], part but not all of item in [3]	Put contents of act in [3]
`Ack`	Item in [5], not in [6]	Move item from [5] to [6]
`Repair`	Either Item in [2] or [3] doesn't match the appropriate item in [1] or Item in [4] is unclear (either no interpretation, no preferred interpretation, or interpretation doesn't match expectations) but there is enough context to say what it *should* be	Delete indicated part of item, move replacement content to [3]
`ReqRepair`	Item in [4] is unclear (either no interpretation, no preferred interpretation, or interpretation doesn't match expectations)	Add Discourse Obligation for Y to `repair`
`ReqAck`	Item in [3] matches item in [1], Y has passed up a chance to acknowledge	Add Discourse Obligation for Y to `ack`

Table 3.3: X's actions

Action	Effects
`Initiate`	Put contents of act in [5]
`Continue`	Put contents of act in [5]
`Acknowledge`	Move indicated item from [3] to [6]
`Repair`	Delete indicated part of act in [3] or [5], move replacement content to [5]
`ReqRepair`	Add Discourse Obligation to `repair`
`ReqAck`	Add Discourse Obligation to `ack`; `ReqRepair` if unsure what Y wants acknowledged

Table 3.4: Y's actions

3.4 Connecting the Cognitive and FA Sequence Models

Table 3.5 shows constraints on performance by X or Y of the grounding acts. The acts are superscripted with I or R, depending on whether the speaker is acting as the initiator or responder, as in Table 3.1. These constraints are all relative to the knowledge of X, as represented in Figure 3.14.

Actions	Conditions for X	Conditions for Y
$\texttt{Initiate}^{\texttt{I}}$	Item in [1], not elsewhere	none
$\texttt{Continue}^{\texttt{I}}$	Part of item in [3], part in [1]	Item in [5]
$\texttt{Repair}^{\texttt{I}}$	Content in [2] or [3] different from content in [1]	Item in [4] or [5]
$\texttt{Repair}^{\texttt{R}}$	Item in [4] or [5] is incorrect	Item in [3]
$\texttt{ReqRepair}^{\texttt{I}}$	Item in [4] which is unclear	Item in [2] or [3]
$\texttt{ReqRepair}^{\texttt{R}}$	Item in [4] which is unclear, or item in [5] which doesn't seem right	Item in [2] or [3]
$\texttt{Ack}^{\texttt{I}}, \texttt{Ack}^{\texttt{R}}$	Item in [5]	Item in [3]
$\texttt{ReqAck}^{\texttt{I}}, \texttt{ReqAck}^{\texttt{R}}$	Item in [3]	Item in [4] or [5]

Table 3.5: Constraints on Grounding Acts

Using this information, we can now account for the constraints on the distribution of grounding acts in a discourse unit shown in Table 3.1. In state S, there is nothing of the current Discourse Unit in any of the boxes (other than perhaps Box 1), so according to Table 3.5, the only act possible is an `initiate`. Also, an `initiate` act is not possible in any other state, because this DU has already been initiated (though, of course, either party may begin a new DU with a subsequent `initiate` act).

State 1 corresponds to there being something in Box 3 if X is the initiator, or Box 5 if Y is the initiator. From State 1, $\texttt{Ack}^{\texttt{I}}$ is disallowed, because there is nothing in the appropriate box (Box 5 if X is initiator, Box 3 if Y is the initiator) for the act to acknowledge. Similarly, there is nothing for the initiator to request repair of. Continuations and repairs by the initiator will just add more to Box 3 if X is initiator or Box 5 if Y is the initiator. An acknowledgement by the responder will move items into Box 6.

State 2 corresponds to a point after a repair request by the responder. If X is the

initiator, then there is something in Box 3, and the repair request in Box 4. Also, X has a discourse obligation to make a repair. A continuation is precluded because, given the obligation, it would be seen as somehow addressing the request, and therefore a repair. If, somehow, the initiator's next utterance were seen as a continuation, it would be a signal that the initiator did not process the request. As in State 1, the initiator cannot acknowledge or confirm, because there is nothing in the proper box. The expected operation from State 2 is that the initiator will perform the requested repair, but there are a few other possibilities. The initiator might not be able to interpret the request, and may request a repair of the `reqRepair`, shifting the discourse obligation over to the responder, and putting the DU in State 4. The responder may realize that his request might not be interpreted correctly, and may `repair` it, remaining in State 2. He might make a different repair request, also remaining in State 2. The final possibility is that, on further reflection, the responder realizes the answer without the initiator having repaired. In this case the responder may acknowledge the original contribution, just as though the request had never been made. This takes us directly to State F, and removes the obligation to repair.

State 3 is reached when the responder has directly repaired the initiator's utterance. Here the responder is shifting from what the initiator intended for the DU to what the responder intends[5]. In making the repair, an item has been placed in Box 3, when X is responder, or Box 5, when Y is the responder. The responder can be seen as shifting the roles and seizing the initiative. This state is thus a sort of mirror of State 1. The initiator can repair in return, seizing back the initiative and moving back to State 1. Also the responder can make a follow-up repair, adding more items to the appropriate box, but remaining in State 3. The initiator might not understand the repair, and may make a repair request, moving to State 4. The responder might have a problem with something else that the initiator said, and can "release the initiative" with a repair request, moving back to State 2. Also, the initiator can acknowledge the repair, moving the items to Box 6. The responder may no longer acknowledge his own repair, though he may request an acknowledgement, or even rescind the repair (e.g. "oh, sorry, you're right.").

State 4 is perhaps the most complicated state. It corresponds to the responder having a discourse obligation to repair. If we trace back the conditions on the acts, we see this can only happen after an original `initiate` by the initiator, some response by the responder, and then a repair request by the initiator. Thus there is something in each of Boxes 2–5 and the responder has an obligation to make a repair. From this state, the initiator may make a further repair request, or repair her previous request, remaining in State 4, the responder may repair, moving on to State 3, or the responder may request repair of the repair request, moving to State 2.

State F occurs when items have moved on to Box 6. Ideally, things are now grounded, and no further action is necessary. It is still possible, however, to reopen the discourse unit, as shown in the last column of table 3.1. A repair will put the new item in Box 3 if performed by X, and in Box 5 if performed by Y. A `reqRepair` or `reqAck` will produce

[5]Or more precisely what he thinks that the initiator *should* intend.

the appropriate discourse obligation. In addition, a follow-up acknowledgement will keep the DU in State F.

3.5 Deficiencies of the Finite Automaton Grounding Model

While the model given in Section 3.2 has the virtue of simplicity, a question arises as to how accurate this model is: can it cover a sufficient proportion of the grounding phenomena present in natural language conversations? Will the predictions made by the model as to the groundedness of content conform to the intuitions of participants? Will an agent using the model be able to participate naturally in conversation, or will the adherence to the model cause extra confusion or unnatural constraints?

There are a number of phenomena which, at first glance, this simple model does not seem to handle correctly. This section will explore some of those phenomena and consider whether some more complex model might be more appropriate.

multiple acknowledgements While the finite-state model does not preclude multiple acknowledgements as in (8)[6], a fragment from the example conversation in Section 4.3, it does not provide any reason for their occurrence.

(8)

UU Act	UU#	Speaker: Utterance
$cont_9$	9.2	M: move the engine
$cont_9$	9.3	: at Avon
$repair_9$	9.4	: engine E
$cont_9$	9.5	: to
$repair_9$	10.1	S: engine E1
ack_9	11.1	M: E1
ack_9	12.1	S: okay

DU #9 is considered grounded after utterance 11.1. Utterance 12.1 thus does not do any additional grounding work according to this model. So why was it uttered? One possibility is that these extra acknowledgements have to do with Clark and Schaefer's *Grounding Criterion* and *Strength of Evidence Principle*: S felt that the DU was not grounded to a degree sufficient for the current purposes, and so a further acknowledgement was added. Unfortunately, making use of this principle will significantly complicate the analysis, since it requires a notion of degrees of groundedness. If we equate grounding with mutual belief, then we have to have a notion of a degree of confidence in mutual belief rather than a boolean belief operator. Immediately, the question arises as to how many degrees there should be and which degrees will result from particular utterance types. While this is a very interesting area for further research, it is beyond the scope of the present work.

[6]The notation used here is explained in Section 4.3.2. The left column lists the grounding act type, subscripted with the DU it is part of. The middle column shows the utterance number, and the right column, the speaker and the utterance.

center embedding In actual conversation, it seems that we can acknowledge a part of the ungrounded content, whereas the discourse unit model in Section 3.2 only allows the whole discourse unit to be acknowledged at once. Some of the simple cases seem to follow a "stack-based" structure, in which an *insertion sequence* is grounded as part of the grounding of the top-level, as in Clark and Shaefer's contribution model. For these utterances, we'd thus have a structure such as (9).

(9) A: utterance 1
B: utterance 2
A: ack utterance 2
B: ack utterance 1

Some of these sequences can easily be accommodated if utterance 2 starts a new discourse unit. However, if utterance 2 is a repair, and clearly seems to be part of the same unit as 1, things are more problematic. It might seem that a push-down automaton such as that presented in Section 3.1 might be a more appropriate computational model, reflecting this nesting structure. In fact, the example repair fragment presented in example (8), above might well be analyzed as having this structure. Such a model would have second-person repairs and repair requests entering a new *sub-DU* on the stack, which would have to be grounded before the main unit could be.

partial acknowledgement There are also other sequences in which an utterance seems to acknowledge part but not all of the current content, but there is no clear prior separation within the DU of the parts that become grounded from the parts that remain ungrounded. The sequence in (10) illustrates this[7].

(10) 56.2 S: and then which route do you wanna take
57.1 M: took to Avon
58.1 S: yeah
59.1 M: um
59.2 : the shortest route

The repair in 57.1 shows an understanding of part of the previous question, but not quite enough to provide an answer. Even more problematic is the sequence in (11)[8].

[7]Taken from Dialogue 91-6.2 in [Gross *et al.*, 1993].

[8]Taken from Dialogue 91-8.2 in [Gross *et al.*, 1993].

(11) 87.1 M: but if we do that
87.2 : then we won't be able to m /
87.3 : we won't make it
87.4 : the other/ the other constraint won't work either
88.1 S: well
88.2 : you mean
88.3 : t /
88.4 : if / if we do which

The "well" in 88.1 signals acknowledgement (though lack of acceptance of the claim), then 88.4 is a repair request, asking for a more explicit referent for "that" in 87.1.

These sorts of repairs indicate an understanding of the *explicit* material, but a lack of sufficient certainty about the some of the *implicit* contents, such as the destination of the route in example (10), and the precise plan that is the referent of "that" in 87.1 in example (11). While we might be inclined to just treat these clarifications as separate DUs expressing the more detailed information, this would be somewhat problematic, because in the absence of such a clarification, the initiator of the main DU would assume the implicit content was understood after a simple acknowledgement. It would appear, then, that the content of a DU would be sensitive to the type of acknowledgement performed.

3.5.1 Alternative Grounding Models

As a first alternative, consider the model in Section 3.1, in which other-repair, instead of moving to a new state in the DU, actually *pushes* a new sub-DU onto the stack. Moving to such a push-down repair model will have several implications on the general grounding model.

First off, what does one say about the groundedness of a *popped* sub-unit? For example, for the sequence in (12), what would the push-down model claim about whether the material in the repair is grounded? For the finite-state model in Section 3.2, the entire DU will be considered grounded at this point. For a push-down model, there are two options: that the whole DU is ungrounded, or that the repair sub-DU is grounded, but other parts of the utterance are ungrounded.

(12) A init
B repair
A ack

An empirical question is whether sequences like example (12) occur by themselves and are considered grounded. This is a difficult question to settle, since direct second-person repair is fairly infrequent, and even when one does occur without an explicit

second acknowledgement by the repair producer, the next utterance by the repair producer can usually be seen as an implicit acknowledgement. Even if this question were settled on the side of sequences like example (12) being ungrounded, this would not invalidate a finite-state approach, but would merely mean, for example, that the transition from State 3 given an Ack^I should be to state 1, rather than State F. If on the other hand, sequences like example (12) are grounded, then this would be hard to account for in the push-down model.

The big question is whether one actually needs the full flexibility and constraints of a context-free language, or whether a regular language will be sufficient. This question is made more difficult by the fact that human processing limitations will make it unlikely that more than a couple of levels of nesting could occur even with a push-down model – beyond this, the participants would get confused and restart again at the top level. This seems to suggest that a finite-state model will be adequate, though perhaps one might want a few extra states to represent another level or two of nesting.

A push-down model with popped sub-DUs seen as grounded makes an even stronger case that some sort of partial acknowledgement must be happening: often the repaired part does not make much sense isolated from the context of the main DU. This means that if the repaired part is grounded, at least some of the rest of the content must be grounded, perhaps by the initial other repair itself. If we can, in fact, accomplish this partial acknowledgement, then perhaps we can also do it within a more restrictive model as well.

Another way to try to fix the grounding model to accommodate these cases of partial acknowledgement would be to add a notion of *composite acts*, acts which, given an initial DU would split it into multiple DUs, each consisting of a part of the content of the original, and thus the new utterance will function as different acts for the different resulting DUs. Thus, an utterance might acknowledge part of an open DU while repairing another part. For instance, we might crudely represent the relevant state of the conversation in example (11) after utterance 87.4 as (13).

(13) DU 1: state: 1
content: ...
Claim(M,S, IF DO({M,S},THAT)
$\neg$ MakeIt({M,S}))
THAT= (the x (Plan x) $\wedge$ ...)
...

After 88.4, though we might have something more like (14), where M now has a discourse obligation to repair the referent of THAT. Thus, S's actions result in splitting DU 1 here into two parts, acting as an acknowledgement for one part, and as a repair request for the other.

(14) DU 1.1: state: F
content: ...
Claim(M,S, IF DO({M,S},THAT)
¬ MakeIt({M,S}))
...
DU 1.2: state: 2
content: ...
THAT= (the x (Plan x) ∧ ?)
...

A grounding theory such as this is more complex, since it now it requires one to treat DUs as non-atomic, but it allows a very fine-grained analysis of the grounding process. For each bit of content represented in a particular DU, as each utterance is processed, one must decide how the utterance affects the particular bit of content. If an utterance affects different parts of the content differently, then one must split the new DU appropriately. If multiple bits are affected similarly, and the same future action might also affect these bits similarly, then there is good reason to keep them in the same unit. This issue is taken up further at the end of Chapter 7, in which a theory of grounding based on plan execution is presented, allowing formalization of both the grounding acts presented earlier in this chapter as well as the fine-grained "splitting" acts considered in this section.

One question relevant to the decision on whether to use a push-down model is whether some parts *cannot* be grounded until other parts are. If there are such constraints, then this would be a good reason to have a pushdown model. If there are not such constraints, then there really isn't a good reason for such a model.

3.6 Chapter Summary

In this chapter, we have presented a protocol for grounding in conversation that is based on performing *grounding acts* to build up grounded content in the form of *discourse units*. This protocol solves the problems with Clark and Shaefer's *Contribution* model, discussed in Section 2.6. The infinite regression of acknowledgements is avoided by allowing some autonomous acknowledgement acts that do not require further acknowledgement. The protocol allows an agent involved in the conversation to judge whether a unit is grounded and if not what else is required. In addition, the model is finite-state rather than recursive, allowing for a simpler theory and easier on-line processing. A mentalistic model of grounding was also presented that shows the effects on the mental state of the participating agents of processing grounding acts and provides reasons for producing grounding acts, based on linguistic goals. This model is also compared to the protocol, and helps explain why the details are as they are. Finally, some potential problem areas for the finite-state grounding protocol are considered.

Chapter 4 shows how grounding fits in as just one of several levels of action that occur in a conversation. Chapter 5 extends the mentalistic model in this chapter to

a fuller theory of dialogue management, showing how grounding can be naturally fit into a task-oriented natural language conversation system. Chapter 6 describes an implementation of the ideas in the previous chapters as part of the TRAINS system.

4 An Overview of Conversation Acts

Chapter 3 presented the foundation for a computational theory of grounding, deciding for sequences of grounding acts which material was grounded. A cognitive model of grounding was also presented, which showed the rationale for performing grounding acts as well as their effects. There are still several pieces missing from a complete grounding theory, however. First, while the *content* of grounding acts was mentioned, this content, which is the aim of grounding, was left unspecified. In addition, while the cognitive model laid out the rationale for performing one act instead of another depending on the context and goals, it had nothing to say about when to perform an act and when to wait for the other conversant to act. Finally, something must be said about how to recognize the performance of particular acts in a given context. Examination of these issues must involve issues beyond just the grounding phenomena, including issues of turn-taking and utterance content.

This chapter describes the theory of *Conversation Acts*[1], a multi-stratal theory of action in conversation, which includes a level of *Grounding Acts*. Section 4.1 argues for the need of a multi-stratal theory of action. Section 4.2 describes a taxonomy of conversation acts. Section 4.3 presents a short conversation in the TRAINS domain with annotated conversation acts. Section 4.4 describes some of the methods for recognizing conversation acts. Finally, Section 4.5 compares the theory of conversation acts to other multi-stratal action theories.

4.1 The Need for Multiple Levels of Action in Conversation

The models of speech acts described in Section 2.1 are very useful tools for analyzing the relationships between sentential utterances and the mental states of the speaker and hearer of such utterances. We wish to extend this work to be able to model more of the types of coordinated activity that takes place between agents in a conversation than can be modelled using just these sentential-level speech acts.

In order to accomplish this, we have to relax the following assumptions which have usually been made within those theories of speech acts:

[1]Most of this chapter appeared previously in [Traum and Hinkelman, 1992].

1. Utterances are heard and understood correctly by the listener as they are uttered. Moreover, it is mutually expected by both participants that this will be the case.

2. Speech acts are single-agent plans executed by the speaker. The listener is only passively present.

3. Each utterance encodes a single speech act.

Each of these assumptions is too strong to be able to handle many of the types of conversations people actually have, as described in Sections 2.3 and 2.5:

1. Not only are utterances often misunderstood, conversation is structured in such a way as to take account of this phenomenon and allow for repair. Rather than just assuming that an utterance has been understood as soon as it has been said, there is a *grounding* process which builds up to this mutual understanding by including actions by both conversants. The assumption of mutual understanding is not made until some positive evidence is given by the listener (an acknowledgement) that the listener has understood. Some acknowledgements are made with explicit utterances (so-called *backchannel responses* such as "okay", "right", "uh huh"), some by continuing with a next relevant response (e.g. a second part of an adjacency pair such as an answer to a question), and some by visual cues, such as head nodding or continued eye contact. If some sort of evidence is not given, however, the speaker will assume that he has not made himself clear, and either try to repair, or request some kind of acknowledgement (e.g. "did you get that?").

2. Since the traditional speech acts (e.g. Inform, Request) have mutual belief as an effect, they require at least an initial presentation by one agent and some form of acknowledgement by another agent. They are thus are inherently multi-agent actions. Rather than being formalized in a classical single-agent logic, they must be part of a framework which includes multiple agents.

3. Each utterance can encode parts of several different acts. It can be a presentation part of one act as well as the acknowledgement part of another act. It can also contain turn-taking acts, and be a part of other relationships relating to larger-scale discourse structures.

Additionally, there are several different *types* of action being performed in carrying on a conversation. In addition to performing the traditional propositional-level speech acts, conversants are structuring their utterances into coherent segments and also engaging in the turn-taking and grounding processes. While it would be possible to construct individual descriptions of the totality of action being performed, this would miss the generalization that action at these different levels are largely orthogonal to each other. By separating out the action at the different levels, the action descriptions at each individual level can remain more simple.

4.2 Levels of Conversation Acts

We distinguish four levels of action necessary for expressing the content and maintaining the coherence of conversation, as shown in Table 4.1. Action attempts at any of these levels can be signaled directly by surface features of the discourse, although usually a combination of surface features and context will be necessary to disambiguate particular acts. In Table 4.1 the levels from top to bottom are typically realized by larger and larger chunks of conversation, from turn-taking acts, usually realized sub-lexically to grounding acts which are realized within a single utterance unit (UU), to core speech acts which are only completed at the level of a completed discourse unit (DU), to argumentation acts which can span whole conversations. The table also shows some representative acts for each class.

Discourse Level	Act Type	Sample Acts
Sub UU	**Turn-taking**	take-turn, keep-turn, release-turn, assign-turn
UU	**Grounding**	Initiate, Continue, Ack, Repair, ReqRepair, ReqAck, Cancel
DU	**Core Speech Acts**	Inform, YNQ, Check, Eval Suggest, Request, Accept, Reject,
Multiple DUs	**Argumentation**	Elaborate, Summarize, Clarify Q&A Convince Find-Plan

Table 4.1: Conversation Act Types

It is important to note that according to the terminology of Halliday [1961] these classes are *levels* of language description, and not *ranks*. That is, the distinction between these classes is more like that between phonology and syntax rather than that between a word and a phrase. For example, there is no grammar which will build up a grounding act as an ordered collection of turn-taking acts.

4.2.1 The Core Speech Acts: DU Acts

In adapting speech act to spoken conversation, we maintain the traditional speech acts (described in Section 2.1) such as **Inform**, **Request**, and **Promise**, referring to them as *core speech acts*. To model the multi-utterance exchanges necessary for mutual understanding of core speech acts, we posit a level of structure called a **Discourse Unit** (DU). As described in Chapter 3, a DU consists of an initial presentation, and as many subsequent utterances by each party as are needed to make the act mutually understood, or *grounded*. The initial presentation is best considered a core speech act *attempt*, which is not fully realized until its DU is grounded. A minimal DU contains an initial presentation and an acknowledgement (which may be implicit in the next presentation by another speaker). However, it may also include any repairs or continuations that

are needed to realize the act. A discourse unit corresponds more or less to a top-level *Contribution*, in the terminology of [Clark and Schaefer, 1989] (see Section 2.5). The structure of discourse units is described more fully in Chapter 3.

Some of the core speech acts occurring in the example conversation in Section 4.3 and their approximate meanings are given below. This is not meant as an exhaustive list of core speech act types. As in Chapter 3, *initiator* is the agent who makes the initial presentation (often called *speaker* in traditional single agent accounts, e.g., [Allen, 1983b]), and *responder* is the other agent (often called *hearer* in previous accounts) who must acknowledge the presentation for the act to have its expected effect.

inform Initiator presents responder with new information in an attempt to add a new mutual belief.

ynq Initiator asks responder to provide information that initiator is missing but suspects that responder may know; imposes a discourse obligation on responder to respond.

check Like a **ynq**, but initiator already suspects the answer; initiator wants to move the proposition in question from individual to mutual belief, or bring the belief into working memory.

eval An evaluation by the initiator of the "value" of some (physical or intentional) object.

suggest Initiator proposes a new item as part of a plan.

request Like a **suggest**, but also imposes a discourse obligation to respond.

accept Initiator agrees to a proposal by responder.

reject Initiator rejects a proposal by responder.

4.2.2 Argumentation Acts

Higher-level discourse acts may be built out of combinations of core speech acts. For instance, an **inform** act may be used in order to summarize, clarify, or elaborate prior conversation. A very common argumentation act is the Q&A pair, used for gaining information. Also, a combination of informs and questions can be used to convince another agent of something. An agent may even use a whole series of acts in order to build a plan, such as the top-level goal for the conversations in the TRAINS domain [Allen and Schubert, 1991]. The kinds of actions generally referred to as *rhetorical relations* [Mann and Thompson, 1987] take place at this level, as do *adjacency pairs* [Schegloff and Sacks, 1973] and many of the actions signaled by cue phrases.

4.2.3 Grounding Acts: UU Acts

An *utterance unit* (UU) is defined as continuous speech by the same speaker, punctuated by prosodic boundaries (including pauses of significant length and boundary tones). Each utterance corresponds to one *grounding act* for each DU it is a part of. An utterance unit may also contain one or more turn-taking acts (see below). The grounding acts (introduced in Chapter 3) are defined informally below.

Initiate An initial utterance component of a Discourse unit – traditionally this utterance alone has been considered sufficient to accomplish the core speech act. An `initiate` usually corresponds to the (first utterance in the) presentation phase of a top level Contribution in [Clark and Schaefer, 1989].

Continue A continuation of a previous act performed by the same speaker. A `continue` is expressed in a separate utterance unit, but is syntactically and conceptually part of the same act.

Ack An acknowledgement claiming or demonstrating understanding of a previous utterance. It may be either a repetition or paraphrase of all or part of the utterance, an explicit signal of comprehension such as "ok" or "uh huh", or an implicit signaling of understanding, such as by proceeding with the initiation of a new DU which would naturally follow the current one in the lowest level argumentation act. Typical cases of implicit acknowledgement are answers to questions. Acknowledgements are also referred to by some as *confirmations* (e.g. [Cohen and Levesque, 1991a]) or *acceptances* (e.g. [Clark and Schaefer, 1989]). We prefer the term *acknowledgement* as unambiguously signaling understanding, reserving the term *acceptance* for a core speech act signaling agreement with a proposed domain plan.

Repair Changes the content of the current DU. This may be either a correction of previously uttered material, or the addition of omitted material which will change the interpretation of the speaker's intention. A `repair` can change either the content or core speech act type of acts in the current DU (e.g. a tag question can change an `Inform` to a `YNQ`). `Repair` actions should not be confused with domain clarifications, e.g. CORRECT-PLAN and other members of the *Clarification Class* of Discourse Plans from [Litman and Allen, 1990]. `Repairs` are concerned merely with the grounding of content. Domain clarifications, which modify grounded content, are argumentation acts.

ReqRepair A request for a repair by the other party. This is roughly equivalent to a *next turn repair initiator* [Schegloff *et al.*, 1977]. Often a `reqRepair` can be distinguished from a `repair` or `ack` only by intonation. A `reqRepair` invokes a discourse obligation on the listener to respond with either the requested repair, or an explicit refusal or postponement (e.g. a followup request).

ReqAck Attempt to get the other agent to acknowledge the previous utterance. This invokes a discourse obligation on the listener to respond with either the requested

acknowledgement, or an explicit refusal or postponement (e.g. a followup repair or repair request).

Cancel Closes off the current DU as ungrounded. Rather than repairing the current DU, a `cancel` abandons it; the underlying intention, if it is still held, must be expressed in a new DU.

Chapter 3 shows in detail how sequences of grounding acts can be used to build complete DUs, as well as giving motivations for performing particular acts, and their effects on the conversational state and mental states of the participants.

4.2.4 Turn-taking Acts: Sub UU Acts

We posit a series of low-level acts to model the turn-taking process [Sacks *et al.*, 1974; Orestrom, 1983]. The basic acts are **keep-turn**, **release-turn** (with a sub-variant, **assign-turn**) and **take-turn**.

There may be several turn-taking acts in a single utterance. The start of an utterance can be a take-turn action (if another party initially had the turn), the main part of the utterance generally keeps the turn, and the end might release it (or keep it to follow up with another utterance). Conversants can attempt these acts by any of several common speech patterns, ranging from propositional (e.g. "let me say something") to lexical to sub-lexical. Many turn-taking acts are signaled with different intonation patterns and pauses. Although a conversant can attempt a turn-taking action at any time, it will be a matter of negotiation as to whether the attempt succeeds. Conversational participants may engage in a "floor battle" where one tries to keep the turn while another tries to take it. Participants may also use plan recognition on seeing certain kinds of behavior to determine that the other party is attempting to perform a particular act and, if cooperative, may then facilitate it (e.g. refraining from taking a turn when signaled that another wants to keep it, or releasing when another wants to take the turn).

Any instance of starting to talk can be seen as a `take-turn` attempt. We say that this attempt has succeeded when no one else talks at the same time (and attention is given to the speaker). It may be the case that someone else has the turn when the `take-turn` attempt is made. In this case, if the other party stops speaking, the attempt has been successful. If the new speaker stops shortly after starting while the other party continues, we say that the `take-turn` action has failed and a `keep-turn` action by the other party has succeeded. If both parties continue to talk, then neither has the turn, and both actions fail.

Similarly, any instance of continuing to talk can be seen as keeping the turn. Any signals that are part of the speech stream may be seen as `keep-turn` actions; certain sound patterns, such as "uhh", seem to carry no semantic content beyond keeping the turn. Silent pauses are opportunities for anyone to take the turn. "Filling" pauses with such utterances as "uhh" can signal desire to keep the turn through what might otherwise be seen as a `release-turn`. Certain pauses are marked by context (e.g. a previous topic introduction or request) as to who has the turn. Even here, an excessive pause can open up the possibility of a `take-turn` action by another conversant.

Release turn actions are usually signaled by intonation. `Assign-turn` actions are a subclass of `release-turn` in which a particular other agent is directed to speak next. A common form of this is a question directed at a particular individual. Another is naming the next speaker.

Another act, which would be necessary in a face-to-face, or multi-channel communication situation would be `pass-up-turn`. "Back-channel" items such as "okay" and other signals of attention such as gestures are often analyzed together as not taking a turn, leaving the previous speaker in control (e.g., [Yngve, 1970; Duncan and Niederehe, 1974]). Because in our domain-setting all of these items must proceed through the same channel (the audio), and the other speaker often does stop and wait for the response before proceeding on, we analyze these short utterances as a `take-turn` followed quickly by a `release-turn`. The only thing we might classify as `pass-up-turn` would be silence following a `release-turn` or `assign-turn`.

4.3 Conversation Acts in an Example Conversation

We have distinguished four very different communicative functions, and introduced the principal actions that serve each. Now we will show how these communicative functions are realized in our conversation data.

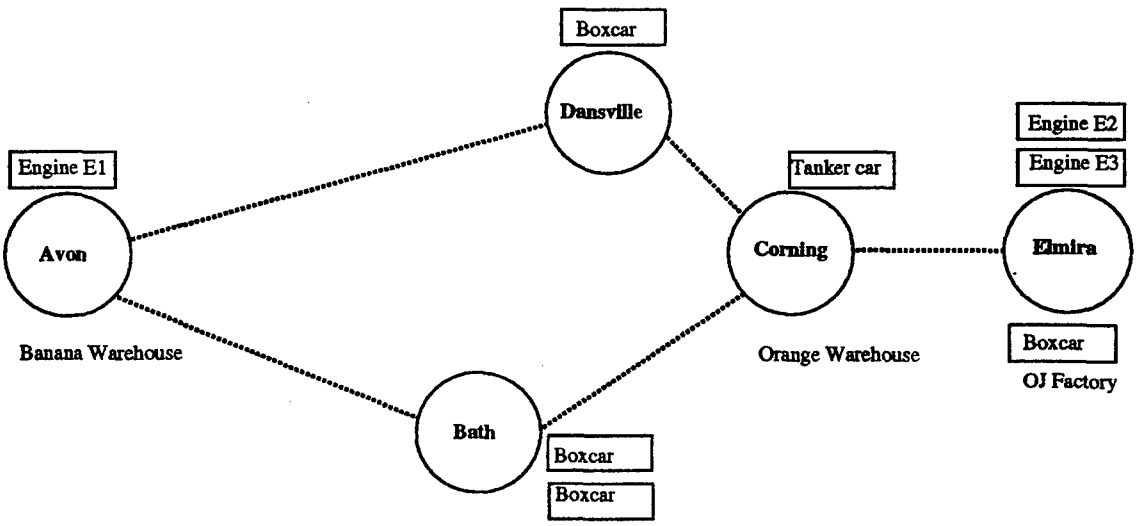

Figure 4.1: Trains World Set-up for Example Conversation

We take our examples from a target corpus of task-oriented spoken dialogues in the TRAINS domain [Gross *et al.*, 1993]. In these dialogues, the manager (M) brings to the system (S) a task to accomplish in a transportation and manufacturing domain. The participants then converse until they have built up a plan which the system believes will satisfy this task. Figure 4.1 shows the initial configuration which the participants must work from, including cities with commodity warehouses and factories connected by rail lines, and train cars which can travel between the cities and carry commodities.

Figures 4.2 and 4.3 show an example conversation[2], with relevant utterance-final features annotated using the Pierrehumbert pitch description system [Pierrehumbert and Hirschberg, 1990][3]. The transcript can be read as follows: the first column lists the utterance number (UU #). The numbering is by turn on the left side of the ".", and utterance within the turn to the right of the ".". Thus UU 15.3 is the third utterance in the fifteenth turn. Principles for the determination of utterance units are given in [Gross *et al.*, 1993]. The next column gives the speaker: "M" for the manager, and "S" for the system. Lack of a marking indicates the same speaker as for the previous utterance.

The Pierrehumbert pitch description system uses two underlying pitch primitives: H (high), and L (low). An intonational phrase is composed from lexical pitch accents (with stressed syllables marked with "*"), a phrase accent at the end of each intermediate phrase, and a final boundary tone designated with "%". There is also a scheme for realization of these underlying pitch features in the utterance. These judgments were made without reference to processed digital signals, so the reader must use them only as a rough guide. We have marked only clearly identifiable features of the utterances; note that the division of the text into utterances often corresponds to a phrase accent but not an utterance boundary tone. We have also annotated final lowering, a less local phenomenon, with dots.

4.3.1 Turn-Taking Acts

Turn-taking acts are difficult to illustrate with a transcript, since they depend heavily on timing and prosodic features. Nevertheless, we will point out a few in the example conversation. `Keep-turn` actions are realized in several different ways in this dialogue. In turn 3, this seems to be the main purpose of the item "now" in utterance 3.1, as well as the utterance "umm" in 3.2. Here, the "now" could also signal a topic shift (from specifying the goal to working on the plan, see Section 4.3.4 below). But this would still be consistent with the fact that M has clearly not thought out what to say next and wants time to work it out rather than letting S take over or permitting awkward silence. In the same vein, the flat end of 3.6 might momentarily be seen as a `release-turn`, but the quick start of UU 3.7 is a clear `keep-turn`. The stretched endings in 15.3, 15.4, and 15.6 all signal keep-turns as well, allowing continuation past both clause boundary and semantic completion (see [Ford and Thompson, 1991]).

`Release-turn` actions are, in the `reqAck` forms 3.8, 5.2, and 7.3, signaled by HH% contours. The turn is also released after the wh-question form in 17.1, and after declarative sentences ending with 1.6, 15.10 and 18.3. Note that in a two-party conversation there is relatively little difference between a `release-turn` and an `assign-turn`, since there is only one potential respondent.

Within each turn, a take-turn action occurs at the beginning of the first utterance (beginning of utterance x.1 for any x). Of special interest is 10.1, in which S takes the turn in order to insert a repair even though there is no prior release-turn from M.

[2]Conversation 91-6.1, from [Gross *et al.*, 1993]

[3]This prosodic transcription and some of the subsequent analysis were done by Elizabeth Hinkelman.

```
UU #  Speaker:Utterance
-----------------------------------------
1.1     M: okay
                H*    H*LL%
1.2      : the problem is
                           H
1.3      : we _better_ .. ship
                          L
1.4      : a boxcar of oranges to Bath
1.5      : by
               %
1.6      : 8 A M
              %
2.1     S: okay
         [2sec]
            *
3.1     M: now
         [2sec]
            *
3.2      : umm
         [5sec]
         L*LL%
3.3      : so
                          *  H%
3.4      : need to get a boxcar
                 *   H
3.5      : to Corning
                    .........*..   H%
3.6      : where there are oranges
                                  H*   L
3.7     M: there are oranges at Corning
           H*HH%
3.8      : right
             L*LL%
4.1     S: right
                         H*                    *  *
5.1     M: so we need an engine to move the boxcar
              H*HH%
5.2      : right
              L*LL%
6.1     S: right
7.1     M: so there's an engine
           ..L*  H
7.2      : at Avon
              H*HH%
7.3      : right
              L*LL%
8.1     S: right
```

Figure 4.2: TRAINS Domain Conversation with Intonational Features – First Part

```
UU #  Speaker:Utterance
-----------------------------------------
9.1     M: so we should
                 . . . .
9.2      : move the engine
               *   H
9.3      : at Avon
                 *H
9.4      : engine E
9.5      : to
                 **HH%
10.1    S: engine E1
           **
11.1    M: E1
           *  *H
12.1    S: okay
                 **H
13.1    M: engine E1
              H*
13.2     : to Bath
13.3     : to /
           H*
13.4     : or
                                          H*                  .........*..*.. H*..LL%
13.5     : we could actually move it to Dansville to pick up the boxcar there
            * * H
14.1    S: okay
15.1    M: um
15.2     : and
                        H*      .........*    H
15.3     : hook up the boxcar to the engine
                         H*      ......H*.....H
15.4     : move it from Dansville to Corning
15.5     : load up
               H
15.6     : some oranges
              .... H*.   H
15.7     : into the boxcar
15.8     ; and
15.9     : then
           ..............* LL%
15.10   M: move it on to Bath
           * * H
16.1    S:  okay
           *   *    H* ...*  LL%
17.1    M: how does _that_ sound
            *    *          *    ...H*...LL%
18.1    S: that gets us to Bath at 7 AM
18.2     : and
                       ......... LL%
18.3     : so that's no problem
            L*LL%
19.1    M: good
20.1    S: okay
```

Figure 4.3: TRAINS Domain Conversation with Intonational Features – Second Part

This conversation is unusual in having no overlapped speech or floor contention. Other dialogues from this study [Gross *et al.*, 1993] contain many examples of **take-turn** and **keep-turn** acts which fail when the other party does not yield the turn. They also contain failed **release-turns** and **assign-turns** where the floor is not taken up by the other party.

4.3.2 Grounding Acts

Figure 4.4 shows the conversation labeled with the grounding acts which correspond to each utterance. Each act is subscripted with the number of the DU of which it is a part. **Repairs**, **continues**, and demonstration style acknowledgements also have in parentheses the UU which they are most directly connected to.

We can see all three types of acknowledgements in this short dialogue. UU 11.1 is a demonstration style acknowledgement: M saying "E1" demonstrates receipt of the repair in 10.1. UUs 2.1, 12.1, 14.1, 16.1, and 20.1 are explicit acknowledgements, claiming receipt but not demonstrating what they are acknowledging. UUs 5.1, 7.1, 9.1, and 19.1 are implicit acknowledgements, recognizable as initiations of DUs whose contents are next steps after the current DU in argumentation-level acts. Thus 5.1 and 7.1 cover further steps in the domain plan (see Figure 4.6, below), and 19.1 gives a relevant evaluation. UUs 4.1, 6.1, and 8.1 are a middle ground between the paraphrase type (in virtue of the repeated lexeme, though with different intonation) and the implicit type – affirming the checks made in the previous turn.

UUs 3.8, 5.2, and 7.3 are all requests for acknowledgement, making more explicit and intense the discourse obligation to acknowledge the current DU which is normally an implicature of a release-turn after an initiate action. An alternative interpretation for the grounding acts performed by these utterances would be to see them as tag-style **repairs** to the Core Speech Act types in their respective DUs, changing the types from informs to questions.

Initiate can be distinguished from **continue** mainly by context. If there is an ungrounded open DU for which the current utterance forms a syntactic and semantic continuation, the current utterance is seen as a **continue**. The very same utterance occurring after an interjected acknowledgement is an **initiate** of a new DU. Thus UU 15.4 is a **continue**, while UU 15.3 is an **initiate**, though on the syntactic level and the domain plan level both just continue the plan from UU 13.5. Note that if the acknowledgement in UU 14.1 were absent, 15.3 would be marked as a **continue** as well. Notice further that 3.7 is labeled an **initiate**, in spite of DU #2 still being open and ungrounded. This is because of the abrupt change in sentence and speech act between 3.6 and 3.7. The importance of the distinction is this: when we get an acknowledgement, how much is being acknowledged? An explicit acknowledgement such as UU 16.1 grounds the entire DU – the **initiate** in 15.3 and the subsequent continuations in 15.4 through 15.10. It does not ground 13.5, because that is already grounded. Similarly, 4.1 explicitly grounds only 3.7 and 3.8, leaving in question whether 3.4 through 3.6 are grounded (see discussion below).

UU Act	UU#	Speaker: Utterance
$init_1$	1.1	M: okay
$cont_1(1.1)$	1.2	: the problem is
$cont_1(1.2)$	1.3	: we better .. ship
$cont_1(1.3)$	1.4	: a boxcar of oranges to Bath
$cont_1(1.4)$	1.5	: by
$cont_1(1.5)$	1.6	: 8 A M
ack_1	2.1	S: okay [2sec]
	3.1	M: now [2sec]
	3.2	: umm [5sec]
	3.3	: so
$init_2$	3.4	: need to get a boxcar
$cont_2(3.4)$	3.5	: to Corning
$cont_2(3.5)$	3.6	: where there are oranges
$init_3$	3.7	: there are oranges at Corning
$reqack_3$	3.8	: right
$ack_3\ init_4$	4.1	S: right
$ack_4\ init_5$	5.1	M: so we need an engine to move the boxcar
$reqack_5$	5.2	: right
$ack_5\ init_6$	6.1	S: right
$ack_6\ init_7$	7.1	M: so there's an engine
$cont_7(7.1)$	7.2	: at Avon
$reqack_7$	7.3	: right
$ack_7\ init_8$	8.1	S: right
$ack_8\ init_9$	9.1	M: so we should
$cont_9(9.1)$	9.2	: move the engine
$cont_9(9.2)$	9.3	: at Avon
$repair_9(9.2)$	9.4	: engine E
$cont_9(9.2)$	9.5	: to
$repair_9(9.4)$	10.1	S: engine E1
$ack_9(10.1)$	11.1	M: E1
ack_9	12.1	S: okay
$init_{10}$	13.1	M: engine E1
$cont_{10}(13.1)$	13.2	: to Bath
$cont_{10}(13.2)$	13.3	: to /
$cancel_{10}$	13.4	: or
$init_{11}$	13.5	: we could actually move it to Dansville to pick up the boxcar there
ack_{11}	14.1	S: okay
	15.1	M: um
	15.2	: and
$init_{12}$	15.3	: hook up the boxcar to the engine
$cont_{12}(15.3)$	15.4	: move it from Dansville to Corning
$cont_{12}(15.4)$	15.5	: load up
$cont_{12}(15.5)$	15.6	: some oranges
$cont_{12}(15.6)$	15.7	: into the boxcar
$cont_{12}(15.7)$	15.8	: and
$cont_{12}(15.8)$	15.9	: then
$cont_{12}(15.9)$	15.10	: move it on to Bath
ack_{12}	16.1	S: okay
$init_{13}$	17.1	M: how does that sound
$ack_{13}\ init_{14}$	18.1	S: that gets us to Bath at 7 AM
$cont_{14}(18.1)$	18.2	: and
$cont_{14}(18.1)$	18.3	: so that's no problem
$ack_{14}\ init_{15}$	19.1	M: good
ack_{15}	20.1	S: okay

Figure 4.4: Conversation with Grounding Acts

The status of UU 9.4 is also somewhat controversial – relying mainly on one's theory of sentence syntax and a judgment about whether 9.4 is repairing a (potentially) inadequate referring expression ("the engine at Avon") in UUs 9.2–9.3, or the three utterances together form a complex referring expression, giving both name and location. It seems that certain complex syntactic phenomena such as asides, vocatives, tags, and left and right dislocation might potentially be seen as separate acts which are really connected only by conversational structure, but a detailed proposal is beyond the scope of the present work (but see also [McCawley, 1988, pp. 763-766] and [McCawley, 1989] for a similar proposal).

This conversation has no repair-requests, but a simple example would be if UU 10.1 had had a rising intonation, or were replaced with "engine what?".

UU 13.4 can be seen as a `cancel` of DU #10. UU 13.5 confirms this interpretation by listing a replacement suggestion. The suggestion in 13.1–13.3 to move Engine E1 to Bath is abandoned and left ungroundable. UU 14.1 grounds the suggestion to move to Dansville, not the disjunction of Bath or Dansville. The explicit `cancel` distinguishes DU #10, which is certainly ungrounded from DU #2, which has a more questionable status. Even though UU 3.7 initiates a new DU, it is still possible to ground DU #2 after this point.

4.3.3 DUs and Core Speech Acts

Table 4.2 shows more information about the DUs in the conversation, listing for each DU which agent was the initiator, what were the types of its constituent core speech acts (superscripted with the performing agent), and the UUs which comprise it. Ungrounded DUs (e.g. DU #10) have their UU list concluded with a "*". DU #2's grounded status is questionable (see discussion below) hence the "?*". UUs which were intended to be part of the DU, but which were abandoned (e.g. UU 9.4, 18.2) are listed in parens. That they were abandoned can also be seen in Figure ?? by the UU arguments of their successor acts, which refer to prior UUs.

In DU #1, M both informs S of the designated problem and suggests that this be the goal of the plan they construct. There are several paths by which this utterance can be recognized as a suggestion: one is by inference from S's expectation that M will propose such a goal (see Section 4.3.4). A second is that in a cooperative environment (such as a TRAINS scenario) an assertion of a need can be sufficient to convey the suggestion of addressing it. S's acknowledgement in UU 2.1 grounds the DU and also accepts the suggestion to work on achieving the suggested goal.

DU #2 also contains two core speech acts – a literal `inform` of the obligation and an indirect suggestion of a necessary subaction which can be recognized as such in the context of problem-solving in the TRAINS domain. It is uncertain whether or not this DU is ever grounded. The content certainly ends up in the final plan, but it could have gotten there just as well through DUs #3, #11, and #12. UU 4.1 might be an (implicit) acknowledgement of DU #2 as well as an explicit acknowledgement of DU #3, but this is not certain. Another possibility is that UU 6.1 acknowledges DU #2. The reasoning for this is as follows: UU 6.1 grounds DU #5, claiming to have understood UU 5.1. But

DU#	Init	Core Speech Act types	Included UUs
1	M	inform^M suggest^{M}accept^S	1.1 1.2 1.3 1.4 1.5 1.6 2.1
2	M	inform^M suggest^M	3.4 3.5 3.6 ?*
3	M	check^M ?suggest^M ?accept^S	3.7 3.8 4.1
4	S	inform-if^S	4.1 5.1
5	M	check^M ?suggest^M ?accept^S	5.1 5.2 6.1
6	S	inform-if^S	6.1 7.1
7	M	check^M ?suggest^M ?accept^S	7.1 7.2 7.3 8.1
8	S	inform-if^S	8.1 9.1
9	M	suggest^M accept^S	9.1 9.2 9.3 (9.4) (9.5) 10.1 11.1 12.1
10	M	suggest^M	13.1 13.2 13.3 13.4*
11	M	suggest^M accept^S	13.5 14.1
12	M	suggest^M	15.3 15.4 15.5 15.6 15.7 15.8 15.9 15.10 16.1
13	M	request^M	17.1 18.1
14	S	inform^S eval^S accept^S	18.1 (18.2) 18.3 19.1
15	M	eval^M	19.1 20.1

Table 4.2: DU Acts from Conversation

UU 5.1 uses a definite reference "the boxcar" which is not licensed by non-linguistic context (the map in Figure 4.1 shows more than one boxcar, any of which would be fine in a plan to move oranges to Bath). The only thing that licenses this use is the mention of "a boxcar" in UU 3.4 – the boxcar which has to get to Corning to get the oranges. So by claiming understanding of UU 5.1, S is indirectly claiming understanding of UU 3.4, and thus grounding at least that part of DU #2 (if it is not already grounded).

The surface actions in DUs #3, #5, and #7 are all clearly **check** actions in spite of their surface declarative form, in virtue of the respective knowledge preconditions [Bunt, 1989; Beun, 1989]. A **check** differs from an ordinary yes-no question in that for a yes-no question the initiator does not know the answer, whereas for a **check** the initiator does know and is making sure both conversants are in synch. A **check** is also of the *question* type rather than the *inform* type, as it requires positive confirmation, not just neutral acknowledgement (replacing UU 4.1 with "okay" does not satisfy the discourse obligation, and replacing it with "oh" – which signals a change in information state in the "oh" producer (see [Heritage, 1984]) violates the presupposition behind the utterance that this information was known by the responder).

We can see by looking at Figure 4.1 (which both conversants had independent access to) that the contents of DUs #3, and #7 were already privately known. The knowledge in DU #5 is background knowledge of the physics of the TRAINS world which could conceivably be unknown to a novice M, but it would surely be known to S, and the confident tone of voice does not indicate a **ynq**. Given that the background knowledge and prosodics already indicate a check for each of UUs 3.7, 5.1, and 7.1–7.2, it is unnecessary to regard UUs 3.8, 5.2, and 7.3 as repairs, as we might do if we had initially considered these DUs to contain **informs**.

In addition, DUs #3, #5 and #7 might have indirect suggestion readings. It is consistent to regard UU 3.7 as suggesting which oranges to use in the plan (although this could have been done previously with UU 3.6 or subsequently with 15.6). DU #5 could be suggesting using an engine to move the boxcar. It is also consistent to regard DU #7 as suggesting the engine to use, though this is made more explicit in DU #9. If we did not have intonation to mark these acts as checks, and since the inform possibilities are ruled out by the domain knowledge, these suggestions would be our only likely alternative (see [Traum, 1991] for examples in another dialogue where this is indeed the case). As it stands, we can not be sure exactly what was meant or understood, though it doesn't matter for the success of the conversational goals.

UU 13.1 in DU #10 clearly begins a suggestion, a follow-up to that in DU #9, but it is cancelled and does not appear in the final plan.

DU #12 ends up as one big compound suggestion which is acknowledged but *not* accepted with UU 16.1. That this is not an accept (as UUs 2.1, 12.1, and 14.1 are) is indicated by the lengthening of the second syllable which suggests a lack of commitment. That M also interpreted it this way is suggested by his follow-up request in UU 17.1, explicitly asking for evaluation of this proposed plan.

4.3.4 Argumentation Acts

Unlike the other classes of acts described here, argumentation acts build up hierarchically within the same class. At the high end, argumentation acts are mainly derivative of the domain task structure – what the conversation is being used by the participating agents to do. At the lower end, cross-domain rhetorical practices are more common – it doesn't matter so much what the task is, there are standard conventional means of achieving it.

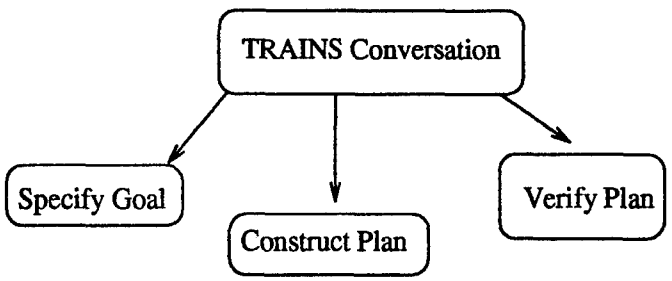

Figure 4.5: Top-Level Trains Conversation Plan

The top-level argumentation acts in the TRAINS Conversations are shown in Figure 4.5. The Manager must first specify the goal, then the participants must construct the plan and the System must verify that the plan meets the agreed upon goals. While, due to different initial knowledge, the Manager is primarily responsible for specifying the goals and the System for verifying the plan, all parts of this process must be grounded to the satisfaction of both parties for the conversation to proceed and conclude successfully. The initiative for constructing the plan is left unspecified by the domain, although

in the conversations collected in [Gross *et al.*, 1993] the System plays a mainly passive role (as in this dialog), offering his own suggestions only when asked or when a problem arises.

We can see that our sample conversation breaks very neatly along this top-level division: DU #1 is the goal specification, DUs 2–12 are concerned with constructing the plan, and DUs 13–15 are concerned with verifying the plan. That things turn out so neatly here is mainly an artifact of the simplicity of the goal in this problem. For more complex problems the breakdown is more often by a vertical decomposition, where a subgoal is specified and then the plan to achieve that subgoal is constructed and verified at which point the process is repeated with another subgoal. Also the steps of constructing and verifying the plan even within a specified subgoal are often intermixed. Poesio [1991] presents a slightly different top-level decomposition based on this tighter coupling.

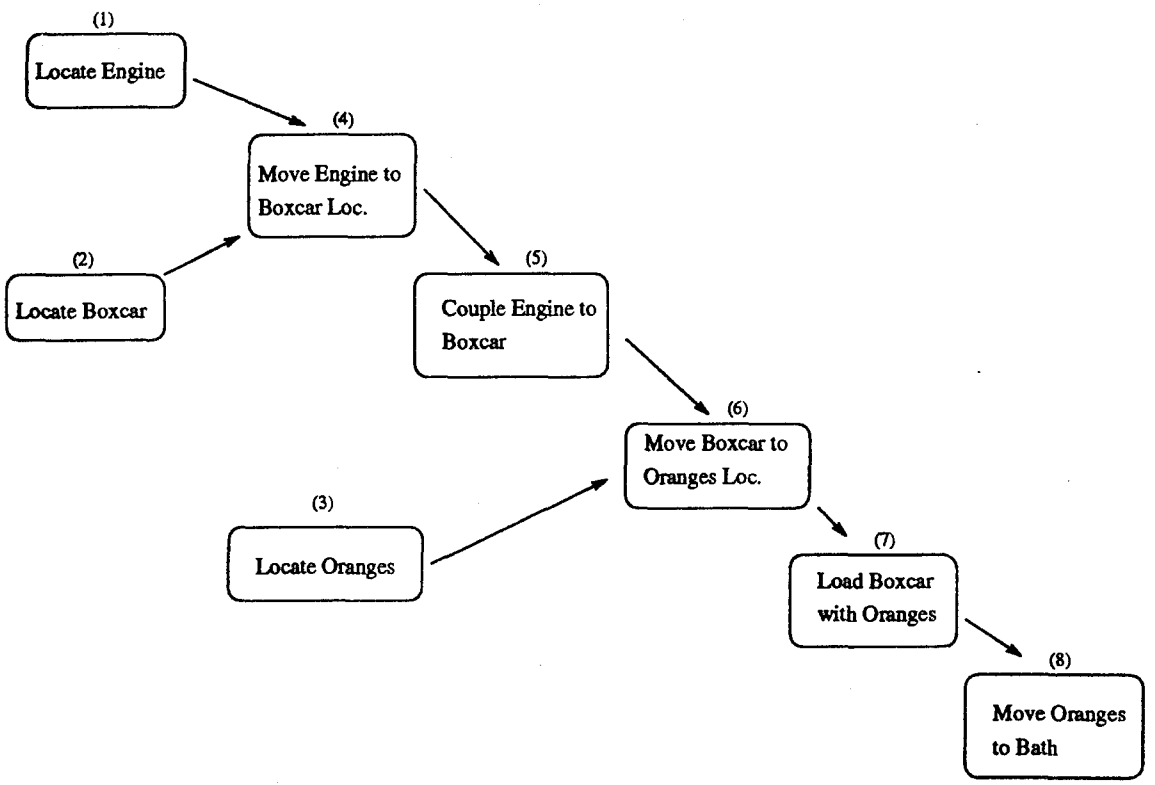

Figure 4.6: Domain Plan for Moving Oranges to Bath

Figure 4.6 provides a typical plan decomposition for this problem (shipping a boxcar of oranges to Bath) in a situation such as this one in which boxcar and oranges and engine are all initially in separate locations. The actions in the plan are numbered arbitrarily for ease of reference, and the arrows represent enabling dependencies in the performance of the actions. For example Action (5), coupling the engine to the boxcar, cannot be performed until the engine has been moved to the location of the boxcar, which in turn cannot be performed until the location of the boxcar and the engine have

been identified. Of course, we may talk about these actions in any of a number of different orders, not just the order of eventual execution.

We can track the connections between the various suggestions in the conversation rather transparently using this abstract plan recipe. The conversational references to the abstract domain plan recipe are shown in Figure 4.7. The numbers beneath the actions represent the DUs in which these steps are described. Thus, for example, action (4), moving the engine to the boxcar location is mentioned in DUs 9, 10 (which is left ungrounded), and 11. Action (5) is referred to in DUs 11 and 12, specifically in UU 15.3 of DU 12. The TRAINS World set-up in Figure 4.1 is repeated here as Figure 4.8.

DU #2 mentions action (6), specifying as well, the location of the oranges (3) but not the particular boxcar (2). As said above, whether this act is grounded or not is not clear, because M immediately goes on to ground (3) explicitly in the Q&A argumentation act consisting of DUs #3 and #4 together. Then DUs #5 and #6 ground the enablement link from Action (1) to Action (6) (shown only indirectly in Figure 4.6 through Actions (4) and (5)). DUs #7 and #8 ground the location of the Engine (part of (1)), and DU #9 elaborates this identification, naming the engine as well. The domain planning behind these utterances has finally gotten to the deepest points in the dependencies, and thus the first steps in actual execution. DU #9 also begins to mention Action (4), but before the destination can be established (UU 9.5 begins to suggest a destination), the repair of the Engine name, UU 10.1, occurs and this needs to be grounded. DU #10 and its replacement #11 elaborate on DU #9, filling in (2), and completing (4). DU #11 then goes on to suggest (5). DU #12 continues the elaboration, repeating the suggestion of (5) in UU 15.3, and then going on in the rest of the turn to suggest (6) (which may have already been suggested in (2)), (7), and (8), completing the plan construction phase.

The plan evaluation phase is begun by DU #13, which is a request to evaluate the whole plan in DUs #3 through #12, referred to by the strongly intonationally marked "THAT" in UU 17.1. This plan is to be evaluated in terms of satisfying the goal set forth in DU #1, as can be determined with access to the top-level decomposition in Figure 4.5. The system's first reply, UU 18.1 is a calculation of the time this whole plan will take. The timings were never brought out in the conversation, but S has the additional information that it takes 3 hours to go from Avon to Dansville, almost no time to couple to the boxcar, 1 hour to move to Corning, 1 hour to load the oranges, and 2 hours to travel to Bath. Thus the whole plan takes 7 hours, and with a starting time of midnight that adds up to 7 a.m. as the time of plan completion. This **inform** supports the requested evaluation, and then S accepts the plan and provides the answer to the question with UU 18.3. M expresses his own approval with DU #15, and the participants now have a grounded plan which they mutually accept as satisfying the grounded goal, and the conversation is completed.

At a lower level of argumentation, we have discourse coherence cues for argumentation relations. Q&As for grounding information content are seen in DU pairs #3-4, #5-6, #7-8, and #13-14. DU #11 elaborates on the plan suggested in DU #9, and this is further elaborated in DU #12. We have many **suggest** and **accept** pairs as well, often within the scope of a single discourse unit (e.g. DUs #1, and #11).

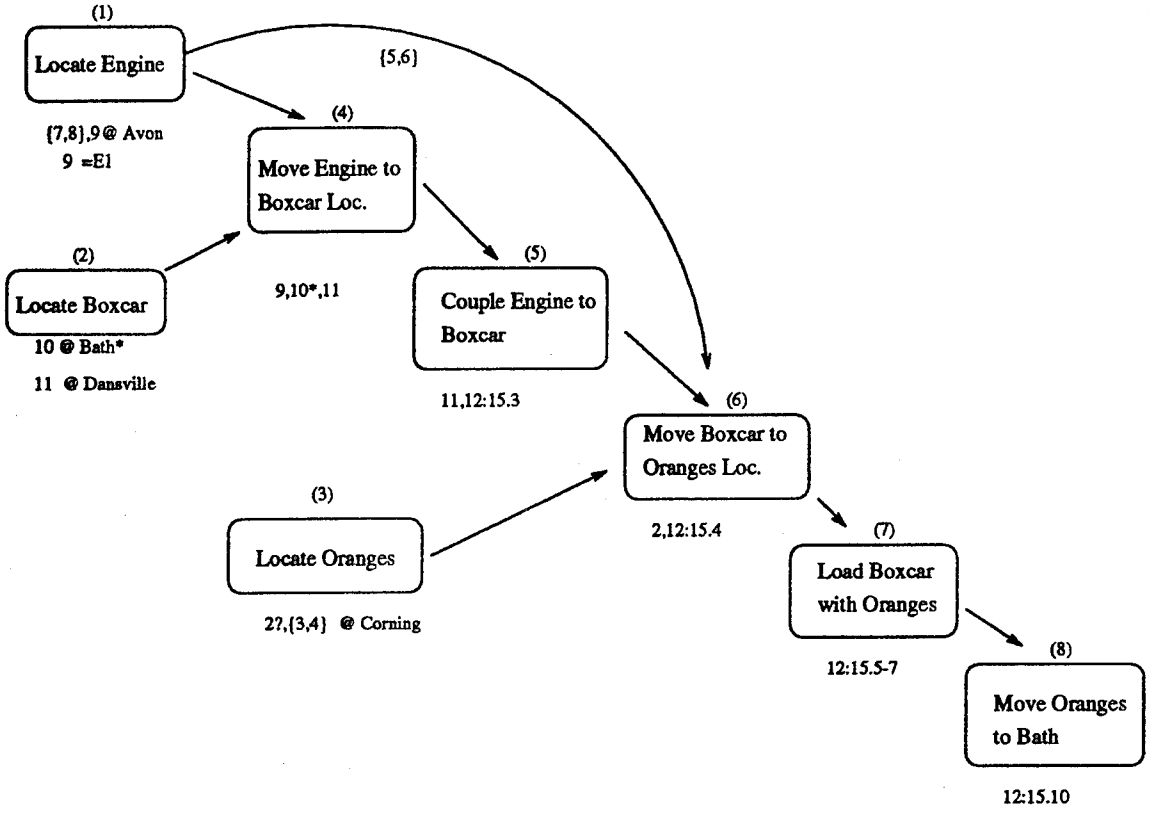

Figure 4.7: Domain Plan for Moving Oranges to Bath

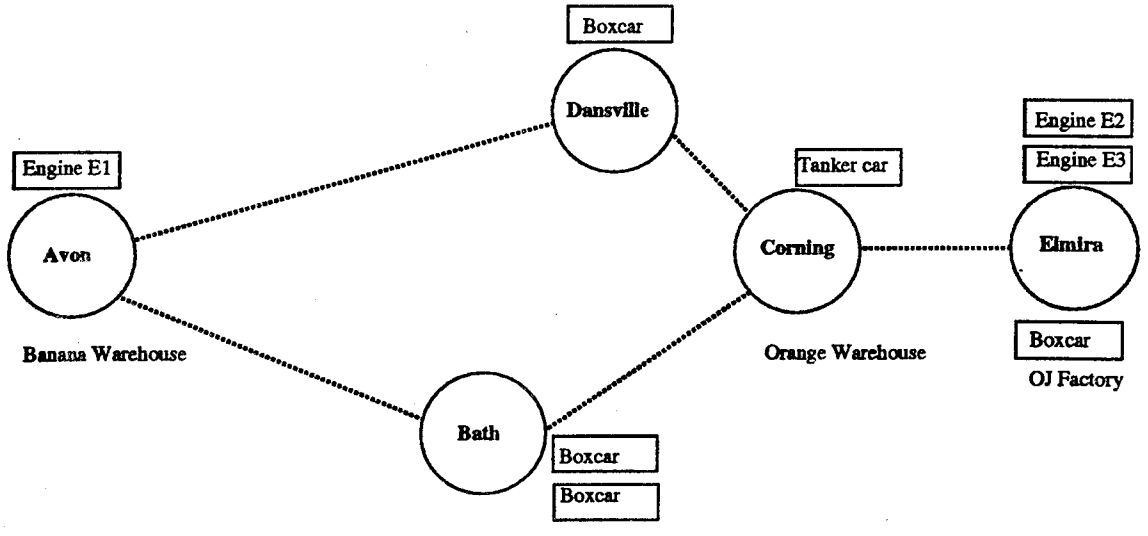

Figure 4.8: Trains World Set-up for Example Conversation

4.4 Recognizing Conversation Acts

Agents participating in discourse are primarily concerned with problem solving, informational, or social goals. The process of meeting these goals through discourse includes recognition of the various conversation acts. Although the recognition process varies according to the class of conversation acts, common elements emerge. For example, each recognition engine can be thought of an evidence combination device, which monitors speech input for certain sets of features. The features serve as evidence for a range of interpretations within the class of acts, but may be overridden by evidence from other features. Also, recognizers may have additional sources of evidence, such as predictions based on previous conversational state. An example of this is the state accumulated in tracking a discourse unit through a sequence of grounding acts (see Section 3.2).

4.4.1 Turn-Taking Acts

Recognition of turn-taking acts is highly dependent upon the social setting and discourse context. Some social settings have preallocated turns or highly formalized turn selection processes. Others have one participant who serves as an arbiter "granting" the turn to a requesting party. In casual conversation, the turn-taking process is generally determined on-line, through use of the same channel that the turn-taking system is regulating.

Two contextual notions are useful for tracking turn-taking in casual task-oriented conversations such as those in the TRAINS domain. These are the *turn* and *local initiative*, described in more detail in Section 5.1.6. Each of these may be said to be held by one (or none) of the participants at any given time in the conversation.

Which agent has the turn is crucially important in recognizing turn-taking attempts, as the turn-taking acts are formulated in these terms. A `take-turn` attempt may only be performed by an agent without the turn, whereas a `release-turn`, and `keep-turn` may only be performed by the current turn holder. A `take-turn` attempt can be recognized as any speech by a non-turn-holder. The other acts are more difficult to distinguish.

Certain indications signal either a `keep-turn` (e.g. filled pauses, lengthened words, pauses at non-constituent boundaries, and "continuing intonation") or `release-turn`. Local initiative is important in determining whether a neutral utterance ending is seen as a `keep-turn` or `release-turn`. Thus a question will impose a discourse obligation on the listener, and although the current speaker may retain the turn and follow the question with, for example, a clarification, the next neutral ending will be seen as a release-turn. In an analogous way, if the current speaker ends an utterance without explicit signals when she is still expected to speak, e.g. after an introduction or in the middle of a list, then this is still keeping the turn. In conversations such as those in the TRAINS corpus, where verbal acknowledgements are important for grounding the content, there is always a mild obligation on the listener to respond, and thus it takes an explicit continuation signal (either intonation or content) to prevent a neutral ending from releasing the turn.

4.4.2 Recognizing Grounding Acts

Recognition of Grounding Acts is highly dependent on the local linguistic context, as described in Chapter 3. Only certain acts will be possible for an agent to perform in a given state of the conversation, and the same utterance will be interpreted differently based upon its surroundings.

In a situation in which there are no accessible DUs, the only possible grounding act would be an `initiate` - thus any utterance that attempted to change the mutual beliefs of the agents would be an `initiate`. `Initiate` can be recognized in other locations as an utterance which conveys new content to be mutually believed which is not a syntactic or semantic continuation or correction of an extant unacknowledged unit. A syntactic/semantic continuation of a unit which has not been acknowledged will be seen as a `continue`. The difference is this: items which are grouped together as part of the same discourse unit will be acknowledged together and those which are not (i.e. the second utterance is marked as a new `initiate` rather than a `continue`) have the option of having one but not the other acknowledged.

Acknowledgements come in three types. "Backchannel" responses (e.g. "okay", "uh huh") in the proper context (state 1 for the Responder or 3 for the Initiator) directly signal acknowledgement of the current DU. A paraphrase or completion of the other's sentence may be either an acknowledgement, a `repair`, or a `reqRepair`. Questioning intonation will signal a `reqRepair`, and acknowledgements are distinguished from repairs by having the same or expected content as opposed to a replacement or new content. Implicit acknowledgements can also be recognized by an initiation of a new DU which forms a next step in a current argumentation act. For example, an answer acknowledges the previously initiated question.

A `repair` is any utterance which replaces any of the content of the current DU. This change may be either to the explicit content of a previous utterance or to the presuppositions. `Repairs` by the Initiator are often signaled by cue phrases such as "I mean", "that is", or "I'm sorry". `Cancels` indicate a dropping of the intention to complete the current DU. Signals include, "forget it", "never mind", and dropping an utterance part way through and starting up with something else.

4.4.3 Core Speech Acts

The general model that we assume for core speech act recognition is that of [Hinkelman and Allen, 1989; Hinkelman, 1990]. This work emphasized the role of surface signals in recognition of speech act type, using patterns of lexical, syntactic, and semantic features to generate a set of hypotheses about speech act type. The set of hypotheses was then further filtered, by testing for each the plausibility of inferences which it would license about relevant propositional information in context. If these inferences were in contradiction to the current knowledge state, this would be grounds for elimination. If the remaining speech act interpretations provided an inadequate basis for action, plan-based reasoning [Allen, 1983b] could be invoked. One hope was that this model would have good real-time properties, and the fact that this model is sensitive to surface

characteristics of an utterance makes it a good candidate for recognition in spoken dialogue, where forms may be incomplete or interrupted.

Clause-by-clause analysis enables integration of core speech act recognition into the framework of turn-taking and grounding. The surface-oriented component of speech act recognition examines each input phrase, as well as any larger constituents that syntactic analysis identifies, and attempts to build speech act interpretations for them. If core speech acts are found (and there are defaults for most syntactic phrases), they are subjected to verification as plausible plans, as above. In the simpler clause-by-clause model it is desirable to buffer interpretations until the utterance boundary is found, to be sure that the topmost node does not block clause interpretations. The integrated model must also rely on utterance boundaries for this purpose. The final set of interpretations is passed on to DU tracking (see Section 3.2) for preservation across utterance boundaries.

Clause-level tracking is useful immediately at the beginning of our example dialogue. Consider the system's view of turn 1. Ignoring the initial "okay", clause-level analysis yields an `inform`, and a declaration of a problem can be a directive act as well. The idiomatic "<person> better" is a forceful suggestion, and unifies with the directive interpretation to yield an overall suggestion act for the utterance. Testing against propositional information in context supports this conclusion.

Consider our example dialogue from the system's point of view, beginning at utterance 3.1. Core speech act recognition gets no interpretation for the first two isolated word utterances, although other modules do. Being declarative, the next clause, in utterances 3.4-3.5 is an `inform`, again an `inform` of a need and therefore potentially carrying the force of a suggestion. Relative clauses, such as "where there are oranges" in utterance 3.6 default to a supplementary `inform`. Intonationally, this turn is a series of phrases with H phrase accents; the last is possibly a bit more elongated than the others, and does not have any conclusive fall. It is thus bounded by the next syntactic unit's beginning, and at this point the interpretation for the entire sentence is that of the first clause.

UU 3.7 and 3.8 follow without strong intonational boundaries. UU 3.7 has declarative syntax but ends in a continuation rise, and 3.8 resolves the interpretation to a confirmation question, or "check". This interpretation is easily verified as a plausible plan, since both speaker and hearer have written copies of the information contained in the utterance.

4.4.4 Argumentation Acts

Argumentation act recognition starts with reference to *Discourse Scripts* [Poesio, 1991]. These represent conventional knowledge of such things as adjacency pairs (e.g, Q&A, greeting–greeting), discourse obligations, and high-level conversational tasks such as those represented in Figure 4.5. These scripts represent background assumptions of the expected coherence relations, which can be applied to the current situation to allow recognition of probable speaker intention. First parts of discourse scripts will make their next parts conditionally relevant, for instance, the first utterance after a question

will be seen as an answer unless it gives explicit signals to the contrary (e.g. a repair or follow-up question).

The main surface indications of argumentation relations are certain types of cue words. "So" is a very good signal for a summary or deduction function. These generally cue items which, while they haven't appeared explicitly in the previous conversation, are inferable from what has gone on before combined with background knowledge. Thus, the **inform** in DU #2 is in a summary relation to DU#1, in virtue of the decompositional plan knowledge that there is only one orange source and a boxcar is needed to carry oranges. Similarly for the **check** in DU #9. The "so" in DU #5 seems like it could be either a summary or just a topic progression, another common use of "so". "And" is a good signal of elaboration or continuation, e.g. UUs 15.2, 15.8.

Domain Plan knowledge will also be extremely important at this level. Often it is not necessary to know the precise relationship between two segments of conversation. As long as the conversants are attending to building up the domain plan and tracking how the contents fit together, that is enough to get by without explicit identification of the rhetorical relationships. Knowing these relationships, however, can provide important clues to the domain plan recognizer as to how to fit a new item into the plan.

4.5 Related Classification Schemes

There have been quite a few previous attempts to categorize acts in discourse into different groups. we believe, however, that none of the previous classifications have the range of coverage that the current scheme has. Most schemes either treat only one or two of the levels we have here, or try to combine all the acts into a system of rankings, where one group is composed of items at a lower rank, in the way that grammatical phrases are composed of words. An example is the classification scheme proposed in [Sinclair and Coulthard, 1975] and later modified in [Coulthard *et al.*, 1981] and [Stenstrom, 1984]. This taxonomy is one of ranks within the same level, a level called "discourse", with the following ranks from smallest to largest: act, move, exchange, sequence, transaction. This system corresponds most closely to the argumentation level, although the exchange rank is very similar to a DU in our terms, consisting of an initiation possibly followed by a response and feedback. Grounding and Turn-taking are not explicitly covered, although there are acts such as *acknowledge* and *reply* in [Coulthard *et al.*, 1981] and *repeat, backchannel, request acknowledgement* in [Stenstrom, 1984].

The taxonomy most closely related to the present one is that of Novick [1988]. Novick, realizing that the traditional speech acts are insufficient for modelling dialogue control, introduces several levels of what he calls *meta-locutionary acts*, to contrast with Austin's locutionary, illocutionary, and perlocutionary acts. The meta-locutionary acts include levels for turn-taking, repair of mutual models, reference/information, and attention. While attention and reference are not addressed in our taxonomy (reference is treated as part of the content of core-speech acts), there is no correlate in Novick's taxonomy of the argumentation acts. While some aspects of grounding are treated at Novick's *repair of mutual models* level, other aspects are left for other levels. Following

Bach and Harnish [1979], Novick treats acknowledgements as a class of speech acts to be found at each level, rather than a specific type of grounding act. There is also no correlate in the mutual models level of the information-providing *initiate* and *continue* acts.

Although the levels of action we have discussed in this chapter are all manifested through the same channel of spoken language, the levels represent coordination of different types of activity. Turn-taking coordinates who is in immediate control of the speaking channel and should have the attention of the participants. Grounding coordinates the state of mutual understanding on what is being contributed. Argumentation coordinates the higher discourse purposes that the agents have for engaging in the conversation. Core Speech Acts coordinate the local flow of changes in belief, intentions, and obligations. There is no way to construct one of these acts via some sequence of acts at another level. Each utterance will contain or be part of multiple acts at these different levels.

5 Dialogue Management

This chapter introduces a general theory of dialogue management for natural language conversation and shows how a speech act theory of grounding such as that given in Chapters 3 and 4 fits into such a theory. The *dialogue manager* of a natural language interface system is the component that handles interaction between the user (via language interpretation and generation) and whatever back end or *domain* model the interface system is connected to. The dialogue manager will generally control the functioning of the rest of the system – calling language interpretation, domain functions and language generation – as required by the demands of the dialogue. A rough schematic of the dialogue manager interactions is shown in Figure 5.1.

The dialogue manager will have to maintain some *state* in order to engage in a natural dialogue. This state will include representations of mentalistic attitudes of the system itself, as well as aspects of a user model and a conversational model. This mental and conversational state is described in Section 5.1. In addition, the dialogue manager will need routines to update the state resulting from actions performed by the user as well as actions the system itself performs. Section 5.2 shows how to update the state after the occurance of conversation actions. Finally, the dialogue manager will require a control strategy to decide what to do given its current state. This is discussed in Section 5.3.

Throughout this chapter we will make reference to the implemented dialogue manager in the TRAINS System, described in more detail in Chapter 6. In the TRAINS domain, the purpose of the system is to interactively construct and execute transportation and manufacturing plans in the TRAINS World. The domain model for the conversation system will thus be a model of the TRAINS world and domain functions will involve actions and plans in that world.

5.1 Mental and Conversational State

This section describes the mental attitudes and conversational state which the dialogue manager models in order to understand and engage in natural language conversation.

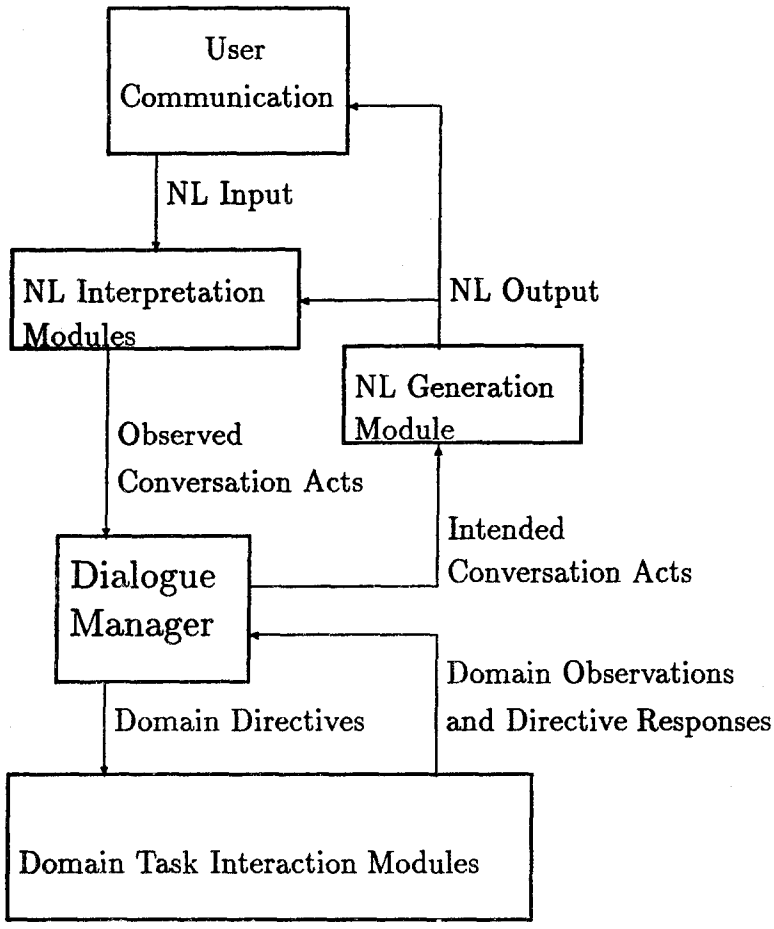

Figure 5.1: Abstract Architecture of Conversation System

5.1.1 Beliefs

Several nestings of domain belief must be tracked in order to fulfill the conversational purposes in a task-oriented dialogue. Private beliefs must be maintained in order to do accurate task reasoning. In addition, the system must maintain beliefs about the user's beliefs and about mutual beliefs both to interpret user utterances and formulate its own expressions coherently. We represent these belief nestings as distinct but related belief modalities.

In addition to actual beliefs about the domain, the system must also track proposals about the domain. Although there will be a natural connection between these and the actual beliefs, there is good reason to keep them distinct during intermediate phases in the conversation. One use is for modelling insincere utterances. Even if the interpretation of an utterance includes a claim that a certain state of affairs holds, there might be good evidence to suppose that the actual beliefs of the speaker are otherwise. Having separate modalities for proposals allows representation of any discrepancy. In addition, this will allow a more explicit representation of the method by which beliefs

change through conversation – the immediate effect of a representational utterance will change only the proposal modality, and it will require an additional (mental) action to actually change the belief. This proposal modality will also be useful for representing suggested courses of events that have not yet been firmly decided upon.

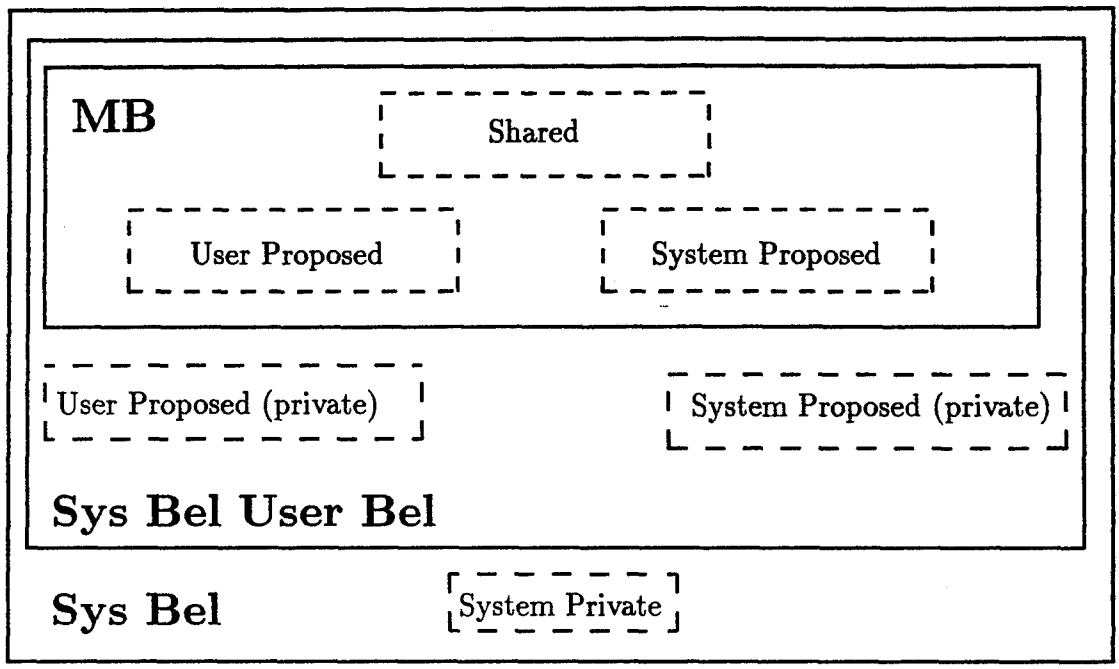

Figure 5.2: Belief and Proposal Contexts

Figure 5.2 shows the relationships between the necessary belief and proposal modalities. The belief modalities are shown with solid boxes, with containment representing the relationships between these modalities. Thus, some of the system's beliefs will be beliefs about the user's beliefs. In turn, some of these beliefs will be mutual beliefs.

Also, some of these beliefs will concern proposals. Part of the mutual beliefs will be mutual beliefs about what the user has proposed and what the system has proposed. Proposals that have been accepted are also shown here as *shared*. For representative proposals (about the actual state of affairs), this *shared* modality will be undistinguished from general mutual beliefs about the world, but for directive proposals (about *actions* to be taken by the conversational participants), the shared context will include an intentional component (to perform these actions and keep them achievable) as well as a mutual belief about future eventualities.

The proposal modalities in the **Sys Bel User Bel** context represent proposals that have been initiated but not yet grounded. Also, in the **Sys Bel** context, the *System Private* modality represents proposals that the system has decided to make but has not yet initiated. Thus, these modalities can be related to the cognitive grounding model in Figure 3.14. Treating the system as the agent being modelled (X), we can see that the *System Private* modality corresponds to Box 1 (X Intends Mutually Believed),

the *System Proposed (private)* modality corresponds to Box 3 (Y believes X Intends Mutually Believed), the *User Proposed (private)* modality corresponds to Box 5 (Y Intends Mutually Believed), and, of course, the MB modality corresponds to Box 6. The modalities within mutual belief do not have to do with grounding, and thus are not represented in Figure 3.14.[1]

For the TRAINS system, many of the utterances will include suggestions of actions, goals, and constraints to add to the current domain plan. From the point of view of the dialogue manager, *domain plans* are abstract entities which contain a number of parts. These include: the goals of the plan, the actions which are to be performed in executing the plan, objects used in the plan, and constraints on the execution of the plan. The detailed structure of TRAINS-93 domain plans and the view of plans as *arguments* are described in detail in [Ferguson, 1992]. The composition of plans is negotiated by the conversational participants throughout the conversation in order to come up with an agreement on an executable plan, which can then be carried out. The conversational participants can have different ideas about the composition of a particular plan, even though they are both talking about the "same" plan.

Views of domain plans are represented as follows. The *shared* modality will include aspects of plans assumed to be shared plans – jointly intended by the system and the user. The mutually believed proposal modalities include plans proposed by one or the other party but which have not yet been accepted. The proposal modalities in **Sys Bel User Bel** represent proposals which have not yet even been acknowledged, and finally the *System Private* modality will contain plans that the system's back-end plan reasoner has constructed but which have not yet been communicated to the user. This framework allows for the representation of both the incremental construction of plans as well as conflicting proposals of what a plan should be, when the plans in different contexts have contradictory components.

5.1.2 Discourse Goals

The system maintains a set of **Discourse Goals** in order to use the conversation to satisfy its own aims. These goals will have a large dependence on the purpose of the entire conversational system and its related task. The discourse goals will generally guide the long term behavior of the system when it has the initiative.

The overriding goal for the TRAINS domain is to work out and perform an executable plan that is shared between the two participants. This leads to other goals such as accepting proposals that the other agent has suggested, performing domain plan elaboration, proposing plans to the other agent that the domain plan reasoner has constructed, or executing a completed plan.

The implemented actor in the TRAINS domain works from a discourse script, which specifies the goals behind particular phases in the conversation. The discourse script for

[1] In considering these correspondences, one might be initially uneasy about the lack of leading *belief* operators in front of the *intends* operators in Figure 3.14. This apparent discrepancy can be reconciled by first remembering that everything in Figure 3.14 is within the context of X's beliefs, and secondly using an axiom of introspection over intentions: $\forall$ A, ϕ: A Intend ϕ $\supset$ A believe A Intend ϕ.

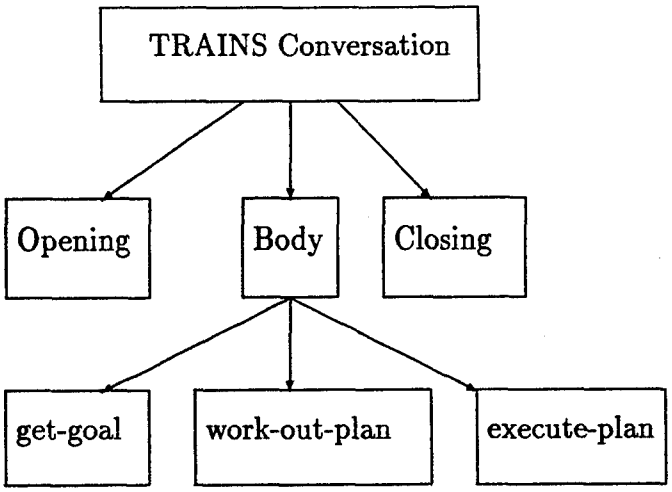

Figure 5.3: Discourse Script for TRAINS Conversations

the TRAINS conversations is shown in Figure 5.3. The system's set of active discourse goals is represented using a stack, with the top goal being primary. When a complex goal is expanded, it is removed from the stack and replaced with a list of its children with the left-most child being on top.

5.1.3 Action

Actions are represented in the reasoning system as semantic primitives. Of particular relevance to the dialogue manager are conversation actions. These are the primary means by which the dialogue manager changes the conversational state and recognizes such changes. Associated with each action type are conditions describing the occurrence and effects on the conversational state of an action of that type. The effects of perceived conversation actions on the discourse state are described in Section 5.2.

5.1.4 Obligations

Obligations are actions that an agent *should* perform, according to external norms. They are different in kind from goals or intentions, though a well-behaved agent will choose to meet its obligations, and in fact there is generally a social cost for not meeting obligations which may encourage a purely strategic agent to endeavor to meet them. In a conversational setting, an accepted offer or a promise will incur an obligation. Also a request or command by the other party will bring an obligation to perform or address the requested action. If these requests are that the system say something (as in an acknowledgement request or a question), then a **discourse obligation** is incurred.

We use these discourse obligations to efficiently and naturally represent the connection between a question and its answer. This can be contrasted with previous systems (e.g. [Allen and Perrault, 1980]), which go through an elaborate plan-reasoning procedure starting from the fact that the question being asked means that the speaker wants

to know something, which should then cause the system to adopt a goal to answer. In the current system, meeting the request is registered directly as an obligation, regardless of the intent of the questioner or the other goal structure of the system. This helps distinguish responses expected by convention (e.g. that a question be answered) from simple cooperative behavior (e.g. doing what one believes another agent wants). Other parts of the system might also bring about discourse obligations, depending on the task, if particular conditions would oblige the system to notify the user. For example, in the TRAINS domain, in some circumstances if the execution of the plan goes wrong, this would bring an obligation to inform the user, who could then help provide an appropriate remedy.

5.1.5 Intentions

As the system discharges its obligations and meets its discourse goals, it constructs a set of **Intended Conversation Acts**, actions it will perform given an opportunity. Because the system attempts to adhere to conventional interactional patterns, it does not always perform these right away, and might not get a chance to perform some of them. For example a suggestion may be preempted by a similar or conflicting suggestion by the user. Also, an answer to a question may become irrelevant if the user retracts the question.

Intentions will help guide the future behavior of the system, generally constraining future planning to fit within the context of prior intention.

5.1.6 Discourse Model

In addition to the mental attitudes described in the preceding sections, there are several aspects of the discourse interactional state that must be tracked. These represent the model of the conversation itself rather than particular mental attitudes of the participants. They are used as a resource to help generate local expectations for the flow of the conversation, as well as serving as an important tool for interpreting subsequent utterances.

The Turn

Two contextual notions are useful for tracking turn-taking in casual task-oriented conversations such as those in the TRAINS domain. These are the **turn** and **local initiative**. Each of these may be said to be held by one (or none) of the participants at any given time in the conversation.

The notion of who has the turn is important in deciding whether to wait for the other agent to speak, or whether to formulate an utterance. It will also shape the type of utterance that will be made, e.g. whether to use some kind of interrupting form or not.

Local initiative[2] can be glossed as providing the answer to the question of who has the most recent discourse obligation – who is expected to speak next according to the default plans for simplest satisfaction of conversational goals. For example, a question or request will produce a discourse obligation for the listener to respond to the request (either by satisfying it, accepting it as a responsibility to be performed later, or denying it). If there are no local obligations, the local initiative may be derived from the higher level discourse goals and expectations.

In the TRAINS domain, the initiative is shared, with different participants holding it at different points in the conversation. In the initial phase, the user has the initiative while the task is conveyed. In the main part of the conversation – the construction of the plan – the initiative can lie with either party, though it generally remains with the user. In the final phase, verifying successful completion of the problem, the initiative belongs with the system.

Grounding Model and Discourse Units

In order to track grounding, the system will maintain a bounded stack of discourse units (DUs), which represents the currently accessible units. For each DU, the system must keep track of the initiator and the *state* of the DU, according to the finite-state grounding theory presented in Chapter 3. In addition, the DU representation will include a list of the (partial) constituent core speech acts and argumentation acts, and will represent their anticipated effects on the discourse context.

Multiple DUs are modelled using a bounded stack structure. The structure is stack-like since, generally, new utterances will affect the most recently started DU. The stack structure is bounded to capture the constraint that DUs have limited accessibility. After enough intervening material, the older DUs are no longer directly accessible (although their content can always be reintroduced in new DUs). Uncompleted DUs which "fall off" the back of the stack are treated as if they had been cancelled – their contents are not considered grounded.

Discourse Segmentation

Discourse Segmentation information [Grosz and Sidner, 1986] is kept for a variety of purposes in linguistic interpretation and generation, including the ability to determine the possible referents for a referring expression and the intentional relations between utterances. The currently open segment structure will guide how certain utterances will be interpreted. Utterances like "yes" can be seen as an acceptance or confirmation of the last question asked but unanswered, if one exists in an open segment. Certain utterances such as "by the way", or "anyway", or "let's go back to" or "let's talk about" will signal a shift in segments, while other phenomena such as clarifications will signal their changes in structure just by the information content. In addition to general segmentation information, a structure of conversationally accessible domain objects is

[2]Roughly the same notion as *Control* in [Walker and Whittaker, 1990], although we use a finer grained notion of utterance types.

maintained. For the TRAINS system, this will include a set of accessible domain plans from a given segment, as well as recency pointers to parts of plans from utterances comprising the segment.

5.2 Updating the Conversational State

The main way that the conversational state is updated is through processing language utterances. The dialogue manager updates the state upon observing any of the various conversation acts described in Chapter 4, as described briefly in the following four sections. Some of the other updates not directly resulting from conversation acts are described in Section 5.2.5.

5.2.1 Updates from Turn-taking Acts

Turn-taking acts will generally only affect the turn. A `take-turn` will make the agent of the `take-turn` action the new turn holder. A `release-turn` will make the agent no longer be the turn holder – in two-party conversation, it will generally make the other agent the new turn-holder. A `keep-turn` action maintains the status of the agent of the action as turn-holder.

5.2.2 Updates from Grounding Acts

Grounding acts will mainly have a local effect on the grounding model. Grounding acts will update the state of the DU of which they are a part, according to Table 3.1. Each grounding act performed as part of an utterance event will also have associated with it a (possibly empty) list of core speech acts and argumentation acts which are attempted in the utterance. While the effects of these acts are only fully realized when the DU of which they are a part is grounded, and the specific effects are described in the next two sections, these acts and their estimated effects are added to the list of constituent core speech acts and argumentation acts for the DU of which the grounding act is a part. `Initiate` and `ack` acts have additional consequences. An `initiate` will add a new DU to the DU stack, often removing an old DU from the bottom of the DU stack. An `ack` will make the contents of that DU mutually believed, thus causing a transfer of contents from the SBUB modality to the MB modality, perhaps causing additional effects such as new discourse obligations or intentions.

5.2.3 Updates from Core Speech Acts

Core speech acts will generally only be fully realized when the DU of which they are a part becomes grounded. The major effects of some of the core speech acts are:

inform The content of the inform is added to the speaker proposed MB modality.

ynq A discourse obligation is added for the hearer to answer or reject the question.

check The content is added to the speaker-proposed MB knowledge base (if it is not already shared). A discourse obligation is added for hearer to respond.

suggest Like an inform, the content is added to the speaker-proposed MB modality. Unlike an inform, a suggestion has an intentional component – accepting it will oblige the recipient to perform some action.

request Like a suggest, but also adds a discourse obligation for hearer to respond.

accept The contents of the accepted act(s) are moved from the proposed MB modality to the shared modality.

reject The contents of the rejected act(s) are not moved from the proposed MB to the shared modality, however whatever discourse obligations to respond to the proposal are removed.

eval The evaluation is added to the speaker-proposed MB modality.

5.2.4 Updates from Argumentation Acts

Argumentation acts will generally affect the assumed coherence of particular core speech acts and other acts and segments. For a suggestion, argumentation acts will help relate this suggestion to the rest of whatever plan is being built. Argumentation acts will also affect the discourse segmentation structure, signaling continuation or completion of particular segments.

5.2.5 Other Updates

Some updates will occur as the result of system-internal "mental" actions. If there is some non-linguistic perception mechanism, observations will affect the system's private beliefs. Changes in belief may also trigger changes in intentions and obligations. In addition, inference may also change the beliefs.

In the TRAINS system, domain plan elaboration will often add new beliefs to the *system private* modality. The domain plan execution and monitoring module may also report changes in the state of the world, leading to a change in the system's beliefs.

5.3 The Discourse Actor

The **Discourse Actor** is the central *agent* of the Dialogue Manager. It decides what to do next, given the current state of the conversational model. It can perform conversation acts by sending directives to the **NL Generator**, and make calls to the domain modules to carry out domain tasks. In the TRAINS system, the dialogue manager can call on the **domain plan reasoner** to perform both plan recognition and plan elaboration tasks and on the **plan executor** to execute a completed plan in the TRAINS world.

The discourse actor must attempt to meet its goals while engaging in rational, cooperative conversation. It must meet its obligations, act on its intentions, and plan and act to meet its goals. Given the mental and conversational state model presented in Section 5.1, there is still a wide range of flexibility for designing a discourse actor. For example, one must set the relative priorities of discharging obligations, performing intended acts, and meeting discourse goals. There are a number of trade-offs possible in these choices, resulting in different styles of behavior. Choosing to act first on intentions or goals might result in more efficient goal satisfaction, but will generally produce a "rude" interactional style. Ideally one would like to prioritize individual attitudes, so that, e.g., a very important goal might take precedence over a fairly weak obligation.

One must also roughly define the *social relationship* that the system should have with the user - should the system lean more towards accepting proposals from the user or towards arguing for what it thinks is best? Which agent should take the initiative at particular points in the conversation? The proper choice will, in large part, be dependent on the domain tasks. For example, in a tutoring system in which novice users are expected to learn particular material, the system will generally take the initiative to guide conversation towards the predetermined goals. In some other types of systems, the user will generally be in control and the system should be responsive to user goals.

There is also a question of the proper ratio of task processing to user interaction. For example, in a very time-critical task, task processing should probably take precedence and the system should interrupt normal task execution and respond to the user only for the most urgent communication. On the other hand, for a task in which the user is assumed to have more task knowledge or reasoning ability, or in which task reasoning is expensive, it may be beneficial to ask for help from the user, and to minimize the independent task reasoning.

The discourse actor implemented in the TRAINS system operates according to the algorithm given in Figure 5.4.

The updating of the conversational state resulting from perceived conversation acts or actions of other modules of the system (as discussed in Section 5.2) is assumed to progress asynchronously with the operation of the discourse actor. Thus, whenever the discourse actor is active, it will first decide on which task to attempt, according to the priorities given in Figure 5.4, and then work on that task. After completing a particular task, it will then run through the loop again, searching for the next task, though by then the context may have changed due to, e.g., the observance of a new utterance from the user. The actor is always running and the system decides at particular points whether to speak or not (according to turn-taking conventions); the system does not need to wait until a user utterance is observed to be invoked, and does not need to respond to user utterances in an utterance-by-utterance fashion.

Lines 2–3 indicate that the actor's first priority is fulfilling obligations. If there are any, than it will do what it thinks best to meet those obligations. If the obligation is to address a request, the actor will evaluate whether the request is reasonable, and if so, accept it, otherwise reject it or, if it does not have sufficient information to decide, attempt to clarify the parameters. In any case, part of meeting the obligation will be to form an intention to tell the user of the decision (e.g., the acceptance, rejection, or

```
(1)  while conversation not finished
(2)      if system has obligations
(3)          address obligations
(4)      else if system has turn
(5)          if system has intended conversation acts
(6)              call generator to produce NL utterances
(7)          else if some DU is ungrounded
(8)              address grounding situation
(9)          else if some proposal is not accepted
(10)             consider proposals
(11)         else if some discourse goals are unsatisfied
(12)             address discourse goals
(13)         else release turn or attempt to end conversation
(14)     else if no one has turn
(15)         take turn
(16)     else if long pause
(17)         take turn
(18)     else wait for user
```

Figure 5.4: Discourse Actor Algorithm

clarification). When this intention is acted upon and the utterance produced, this will discharge the obligation.

Other obligation types are to repair an utterance that is uninterpretable or in which the presuppositions are violated, or to answer a question. In question answering, the system will query the appropriate belief modality and will answer depending on the result, which might be that the system does not know the answer. In most cases, the actor will first merely form the intention to produce the appropriate utterance, waiting for the appropriate chance to actually generate the utterance. In certain cases, though, such as a repair, the system might actually try to take control of the turn and produce an utterance immediately.

For other actions, the system adopts a fairly "relaxed" conversational style – it does not try to take the turn until given it by the user, unless the user pauses long enough that the conversation starts to lag (lines 14–17). When the system does not have the turn, the conversational state will still be updated but the system will not try to deliberate or act.

When the system does have the turn, the actor first (after checking obligations) works from intended conversation acts, calling the generator to produce them if there are any (lines 5–6 of the discourse actor algorithm). Whatever utterances get produced are then reinterpreted (as indicated in Figure 5.1) and the conversational state updated accordingly. This might, of course, end up in releasing the turn. It might not be conve-

Initiator	State	Action
System	1	ReqAck
	2	Repair
	3	Ack
	4	Repair
User	1	Ack
	2	Repair
	3	ReqAck
	4	Repair

Table 5.1: Actions to Perform Based on Ungrounded Material

nient to generate all the intended acts in one utterance, in which case there will remain some intended acts left for future utterances to take care of (unless the subsequent situation merits dropping those intentions).

If there are no intended conversation acts, the next thing the actor considers is the grounding situation (lines 7–8). If there is some DU in the DU stack that is not grounded (i.e. not in the final state), then the actor will adopt an intention to perform an appropriate grounding act, depending on the state of the top DU, as shown in Table 5.1

Note that several grounding acts are not generated in this way. For example, all `initiate` and `continue` actions will come about through other intentions to express particular material, these intentions coming either through obligations (as discussed above), or through other discourse goals, as discussed below. Also, most repairs and repair requests will also come about through other means. The intentions leading to the most common type of repair will come about while interpreting utterances. For example, if an utterance interpretation process results in no plausible interpretations or multiple candidates with equal plausibility, this will lead the system to perform a repair or repair request (depending on whether it was the system's own utterance or the user's utterance). Similarly, if a user utterance is interpretable but it contains presuppositions which are in conflict with system's beliefs, the system will be obliged to `repair`.

The most common case in which examining the grounding situation will cause the system to intend an action will be when the system is the responder and the DU is in state 1 – then the system will simply perform an acknowledgement to ground the unit. This will also be the case when a system-initiated DU is in state 3 (following a user `repair`). Two of the remaining cases, system-initiated state 2 and user-initiated state 4, occur following a user repair request. In this circumstance, the system has the discourse obligation to repair, so these repairs should already be intended by the time the actor considers the grounding state. In two other cases, system-initiated state 1 and user-initiated state 3, the system is waiting for the user to acknowledge, and generally the user will have passed up a chance to do so. Here an acknowledgement request is appropriate to give the user further evidence that this is what the system is waiting for. In the remaining two cases, system-initiated state 4 and user-initiated state 2, the system has performed a repair request and is waiting for a repair from the user. As in

Goal	End test	Action
TRAINS Conversation	*	expand-goal
Opening		greet user
Body	*	expand-goal
get-goal	*shared* domain goal	ask about domain goal
work-out-plan	complete shared plan	call domain plan elaborator
execute-plan	plan is executed	call domain plan executor
Closing		say goodbye

Table 5.2: Actions to Perform Based on Discourse Goals

the previous two cases, generally the user has passed up a turn to repair, so the system should infer that its repair request was unclear and repair this further. Another option would be to cancel this DU and start again with something clearer.

If all accessible discourse units are grounded, the next thing the actor considers is the distribution of content in the proposal modalities (lines 9-10). The system will try to work towards a shared proposal, adding intentions to perform the appropriate speech acts to work towards this goal. This includes accepting, rejecting, or requesting retraction of user proposals, requesting acceptance or retracting system proposals, and initiating new system proposals or counterproposals.

For the actor implemented in the TRAINS system, these proposals will generally involve suggestions about additions to the domain plan that is being constructed. The actor will first look for proposals which are in the mutually believed *user proposed* modality but not in the *shared* modality. If any of these are found, it will add an intention to accept the proposal, unless the proposal is deficient in some way – it will not help towards the goal or the system has already come up with a better alternative. In this latter case, it will reject the user's proposal and present or argue for its own proposal. Next, the actor will look to see if any of its own proposals have not been accepted, requesting the user to accept them if they have been simply acknowledged, or retracting or reformulating them if they have already been rejected. Finally, the actor will check the *system private* modality for any parts of the plan which have not been proposed at all yet. If it finds any here, it will adopt an intention to make a suggestion to the user.

If the system does not need to perform any speech acts to adjust the contents of the proposal modalities, then it checks to see if any discourse goals remain unsatisfied, addressing the most urgent goal that remains (lines 11-12).

For each of the goals shown in Figure 5.3, the actor has an end test which specifies when the goal has been achieved, and an action to perform while in that phase. These tests and actions are summarized in Table 5.2. Complex goals have an end test of "*" meaning that they are completed when their last sub-action has been completed.

For the complex goals, the action to perform is just to expand the goal into its constituent subgoals. The `opening` and `closing` phases are complete when both parties have issued some sort of greeting and dismissal, respectively. The `get-goal` phase is

complete when the participants mutually believe they are working towards some goal. If the user has not already provided one by the time the system gets to this phase, the system will ask for one, since the user is the only party that starts the conversation knowing the goal. The main body of the conversation is the `work-out-plan` phase. It is complete when there is an executable plan in the *shared* modality. Whenever the system has the turn in this phase with no higher priorities, it will call the domain plan reasoner to elaborate the plan, placing the results in the *system private* modality, which will potentially serve as the source for future system suggestions. The `execute-plan` phase is complete after the domain plan has been executed in the TRAINS world. While working on this phase, the actor will call the domain plan executor to execute the plan and report back results.

Finally (line 13), if none of the above conditions hold, the system will just release the turn or try to end the conversation, if it is finished.

When the system does not have the turn it is generally quiescent (the exception being to act on obligations), although if no one has the turn or if the user does not seem to be responding, the system might try to take it (lines 14–17).

5.4 Capabilities of the Dialogue Management Model

The model presented here allows naturally for a mixed-initiative conversation and varying levels of cooperativity. Following the initiative of the other can be seen as an *obligation-driven* process, while leading the conversation will be *goal-driven*. Representing both obligations and goals explicitly allows the system to naturally shift from one mode to the other. In a strongly co-operative domain, such as TRAINS, the system can subordinate working on its own goals to locally working on concerns of the user, without necessarily having to have any shared discourse plan. In less co-operative situations, the same architecture will allow a system to still adhere to the conversational conventions, but respond in different ways, perhaps rejecting proposals and refusing to answer questions. It can handle production and recognition of acknowledgement and repair in a natural and fairly comprehensive manner, and can prompt for them when they are required.

6 Implementation

This chapter presents a detailed description of the dialogue manager and conversation act recognition modules implemented as part of the TRAINS system. The theory presented in the previous two chapters is used by the system to participate in natural language conversations with a human user. A detailed example conversation is also presented. Section 6.1 presents an overview of the TRAINS system. Section 6.2 describes the knowledge representation facilities used by the dialogue manager and adjacent modules. Section 6.3 describes other aspects of the discourse context aside from the belief representations. Section 6.4 describes the implementation of the conversation act theory presented in Chapter 4. Section 6.5 describes actions by the discourse actor which were not presented in Chapter 5. Finally, Section 6.6 presents a detailed trace of the conversation act analysis and dialogue manager's actions on the first several utterances in an example conversation. The trace is concluded in Appendix A.

6.1 TRAINS System Overview

The TRAINS project is a long-term research project to develop an intelligent planning assistant that is conversationally proficient in natural language [Allen and Schubert, 1991]. The TRAINS system helps a user construct and monitor plans about a railroad freight system. The user is responsible for assigning cargo to trains and scheduling shipments, scheduling various simple manufacturing tasks, and for revising the original plans when unexpected situations arise during plan execution. The system aids the user in all aspects of this task by interacting in natural language. In particular, the system typically will perform the following tasks:

- Evaluating suggested courses of action, such as calculating expected completion times, detecting conflicts that might interfere with the actions, and so on;
- Filling in details of the proposed plan that do not require the user's attention;
- Suggesting ways to solve particular subproblems as they arise;
- Presenting and describing the current state of the world and how the proposed plan may affect it, including answering questions from the user;

- Dispatching the plan to the different agents in the world, including the train engineers and factory users;
- Interpreting reports back from the engineers and factory users in order to monitor the progress of the plan and to anticipate problems before they arise; and
- Coordinating the correction and modification of plans with the user.

6.1.1 TRAINS System Modules

Figure 6.1 shows the modules of the 1993 implementation of the TRAINS System, as well as the main flow of communication between modules.

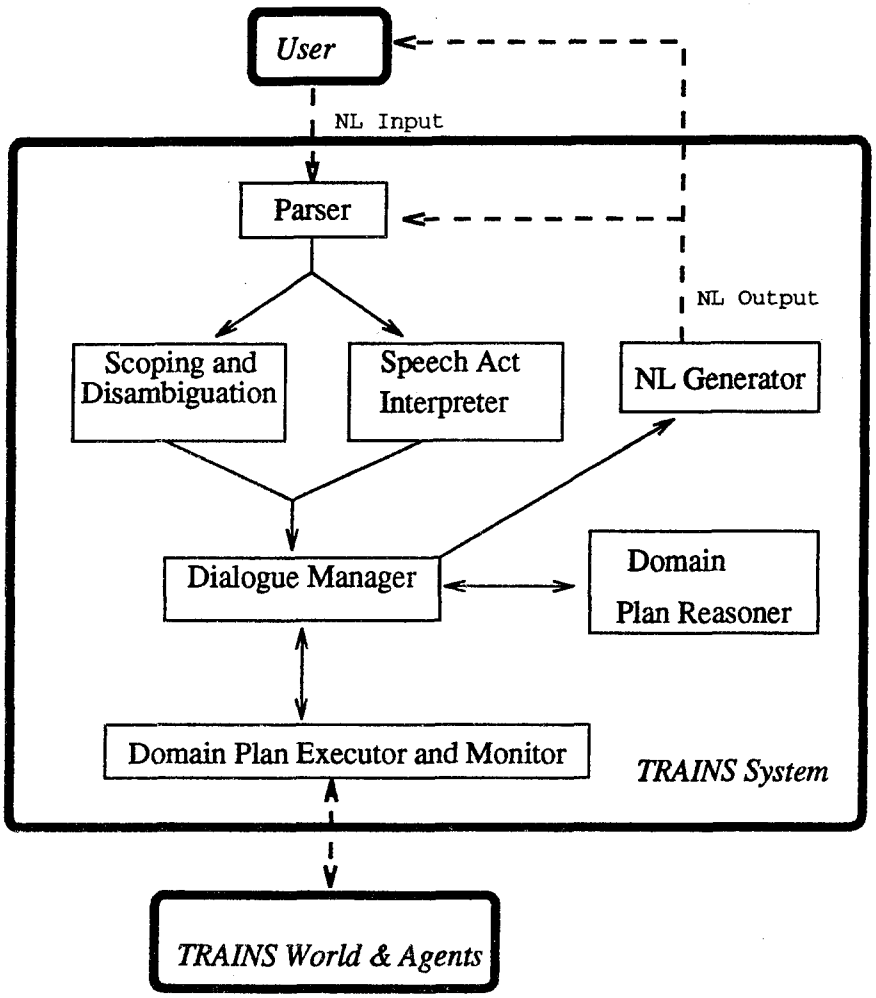

Figure 6.1: TRAINS-93 System Architecture

The system loops between working on perception and action tasks. In the former, the system checks for NL input (from the user or system) to process. If there is any, then the language interpretation modules are called to provide an interpretation which

is used to update the mental and conversational state. While there is no language input to process, the discourse actor is operating, according to the algorithm in Figure 5.4.

6.1.1.1 Language Interpretation Modules

Each utterance (from either the user or the system) is processed by all the modules in Figure 6.1 from the **parser** up to the **dialogue manager** (which calls the domain reasoner to disambiguate hypotheses about meaning and to update the representation of the current plan). The first module in the interpretation process is a **parser**, which takes an utterance and produces a representation that combines the results of syntactic analysis and lexical interpretation. The computed representation is an unscoped logical form (ULF) in *Episodic Logic* [Hwang and Schubert, 1993]. This form is then used for generating hypotheses about the context-dependent aspects of an utterance's content. The **Scope and Deindexing** module makes hypotheses about scope and the objects referred to by referential expressions, and the **Speech Act Interpreter** generates hypotheses about possible speech acts realized by the utterance. These hypotheses are then combined and translated into the *EBTL* language and passed on to the **Dialogue Manager** for verification. The dialogue manager consults the discourse context (described in Section 6.3) and, when plan recognition is required, the domain plan reasoner.

6.1.1.2 Other Modules

The **NL Generator** takes speech acts produced by the dialogue manager and converts them to natural language utterances, which are then output to the user (and also fed back to the NL interpretation modules). The **Domain Plan Reasoner** performs plan recognition and plan elaboration. Plan recognition is a crucial part of understanding utterances of the user, and plan elaboration produces the new information that the dialogue manager will then have to communicate to the user. The **Domain Plan Executor** takes a plan and sends the necessary commands to the individual agents (engineers, factory and warehouse attendants) to have that plan carried out in the simulated world. It also monitors the progress of the plan (by remaining in communication with these agents as they attempt to perform their tasks) to make sure the plan execution is successful.

6.2 Knowledge Representation

The *EBTL* language is used to represent domain plan and conversation level information. EBTL stands for *Event-Based Temporal Logic*. It is a typed logic based on that presented in [Allen, 1991]. It is also used as a communication language between the conversation act analysis, dialogue management and domain reasoning modules of the TRAINS System [Allen and Schubert, 1991]. This section provides an overview of

the syntax, semantics and implemented inference abilities used by the conversation act analyzer and the dialogue manager.[1].

6.2.1 EBTL Syntax and Type Theory

Constants in EBTL are written as a name enclosed in square brackets, e.g. `[ENG3]`. Upper and lower case are not distinguished. Distinct names do not necessarily identify different objects, however. The same object may be referred to by multiple constants.

Variables are notated as `?name*typename` (although subsequent instances of the same variable in the same expression may omit the `*typename`)

Predicates, and quantifiers are written as Lisp keywords (name preceded by a colon, e.g. `:at`, `:the`). Formulae are written as Lisp lists, i.e., in prefix form with parentheses surrounding the expression, e.g. `(:at [Eng3] [CORNING] [T1])`.

6.2.1.1 Types

EBTL is a hierarchically typed language. EBTL follows the RHET [Allen and Miller, 1991] conventions of notating types with the prefix "T-". At the top of the hierarchy is `T-anything`. This type contains all expressions and individuals. Expressions are all of type `T-List`, a subtype of `T-anything`. Individuals are all subtypes of `T-U`, another subtype of `T-anything`.

There is also a distinction made between *simple* types and ***structured*** types. Structured types have associated with them certain functional roles which are inherited by all subtypes.

The top level simple subtypes of `T-U` are: `T-Plan`, `T-Program`, `T-Location`, `T-Object`, and `T-Time`. `T-Agent` is another subtype of `T-U` which is not disjoint with the other subtypes, but has subtypes in common with them. Figure 6.2 shows the type hierarchy for simple domain types. Starred types are also subtypes of type agent.

The top-level structured subtypes of type `T-U` are `T-Event` and `T-State`. States have times as their only roles. Events have both times and locations. `T-Event` is divided into `T-Action` which also contains an agent, and `T-Non-Action` which does not. Speech acts are also subtypes of `T-Action` and are discussed in Section 6.4.

Figure 6.3 shows the hierarchy of structured domain types. Each of these types has a set of associated roles and constraints.

6.2.1.2 Special Predicates

The following predicates are used to bring aspects of the underlying ontology into the domain of discourse of the logic:

---

[1]EBTL was developed in collaboration with James Allen, George Ferguson, and Peter Heeman

```
Location -- City -- Warehouse-City
                    Factory-City
Object ---- Warehouse*
            Factory* ------ OJ-Factory
            Train-Object -- Engine*
                            Car --- Boxcar
                                    Tanker
            Commodity ----- Oranges
                            OJ
            Human*
```

Figure 6.2: Simple Domain Type Hierarchy

```
Event -- Action ------ Load --------- Load-Commodity ---- Load-Oranges
                       Unload ------- Unload-Commodity -- Unload-Oranges
                       Couple
                       Uncouple
                       Move --------- Move-Commodity ---- Move-Oranges
                                                         Move-OJ
                                      Move-Car
                                      Move-Engine
                       Run-Factory -- Run-OJ-Factory
                       Make --------- Make-OJ
                       Bring-About
        -- Non-Action -- Arrive
                       Leave
```

Figure 6.3: Structured Domain Type Hierarchy

- (`:disc-marker` *object1 object2*) is true if *object1* is the discourse marker for *object2*. Discourse markers are used for communication between modules across forms. Discourse markers are constants that refer to intensional objects which may have no realization in the world but are mentioned in conversation. In addition to the `:disc-marker` predicate, EBTL quantifiers have a discourse marker (see below). Currently this predicate is only used where *object2* is a variable, in which case it is always true. Asserting this has the effect of equating the two objects.

- (`:TYPE` *object type*) is true if *object* is of type *type*.

- An equality predicate: (`:eq?` *item1 item2*) is true if *item1* and *item2* are the "same". If the items are individuals, then (`:eq?` *item1 item2*) if the two constants denote the same individual. If the items are complex terms, then they are

`:eq?` if they are unifiable to the same expression.

- A general predicate for testing the relationship between two temporal intervals: (`:Time-Reln` *time1 reln time2*) is true if the indicated relationship *reln* holds between *time1* and *time2*. *reln* can be any of the 13 temporal interval relationships in [Allen, 1983a], or a list of any subset of these intervals, in which case the time-reln expression is true if any of the listed relationships hold.

- A predicate for testing the role values of individuals of structured types: (`:Role` *Item rolename value*) is true if *Item* has a role *rolename* with role value *value*.

- A predicate for testing the actuality of events. EBTL can represent hypothetical events as well as events which actually happen in the world. (`:occurs` *event*) is true if the event actually occurs.

- A predicate for testing the actuality of objects. EBTL can represent intensional constants that (might) have no denotation in the world model. (`:real-object` *object*) is true if *object* denotes an actual object in the TRAINS world.

6.2.1.3 Quantifiers

EBTL includes the following quantifiers:

- An existential quantifier: (`:lf-exists` *var dm rest form*) is true if there is some object, ϕ, (of the type of *var* and which meets the restrictions *rest*) for which, if all instances of *var* in *form* are replaced with ϕ, *form* would be true. *dm* is the discourse marker for the variable *var*. If this form is skolemized, the resulting constant is declared to be equal to *dm*.

- A definite quantifier: (`:the` *var dm rest form*)
  True if there is a unique object, ϕ, (of the type of *var* which meets the restrictions *rest*) for which, if all instances of *var* in *form* are replaced with ϕ, *form* would be true. *dm* is the discourse marker for the variable *var*. If this form is skolemized, the resulting constant is declared to be equal to *dm*.

6.2.1.4 Operators

- A conjunction operator: (`:and` *form**) is true if all *forms* are true

- A complex type constructor: (`:lambda` *var description*) is an abstraction operator of an object, which, when applied to an object, ϕ, will return a copy of *description* in which instances of *var* have been replaced with ϕ.

- (`:apply` (`:lambda` *var description*) *object*) will produce a term of the type of *description* in which all instances of *var* in *description* are replaced with *object*.

6.2.1.5 Domain Predicates and Functions

Domain Predicates:

- The map location of an object at a particular time: (`:at` *object location time*) is true if *object* is at *location* throughout *time.*
- The containment relation for commodities: (`:in` *commodity object time*) is true if *object* is in *object* throughout *time.* A commodity can be in a warehouse or a car.
- The amount of a commodity: (`:quantity` *object amount*) is true if *object* is of size *amount.*

Domain Functions:

- `:loc` maps warehouses and factories to their cities.
- `:boxcar-unit` maps a number to a quantity equal to that many boxcars.

6.2.2 TRAINS Conversation-Level Representation

This section discusses types and predicates used in representing and interpreting dialogue, but not strictly necessary to reasoning about the TRAINS task domain. These include acts and predicates from both the *discourse* and *plan-reasoning* levels described by [Lambert, 1993].

6.2.2.1 Conversation-Level Predicates

These predicates (except `:obligation` and `:weak-oblig`, which have explicit temporal parameters) all refer to the current state of a dynamic conversation. All of them could be seen as having a temporal argument set to the time of evaluation.

- The current focused plan: (`:current-plan` *plan*) is true if *plan* is currently in focus.
- (`:discourse-accessible` *speechact*) is true if *speechact* is part of a discourse unit which is currently accessible (one of the three most recent).
- (`:unaccepted` *speechact*) is true if *speechact* occurs and has not yet been accepted.
- (`:obligation` *agent form time*) is true if *agent* has an obligation at *time* to make *form* true.
- (`:weak-oblig` *agent form time*) is true if *agent* has a weak obligation at *time* to make *form* true. A weak-obligation means that the agent *should* do something as opposed to an obligation which means the agent *must* do it. If obligation were modelled as a set of permissible worlds, then weak-obligation would be modelled as a set of preferred permissible worlds.

- (`:need-for-purpose` *obj form*) is true if *obj* is needed for *form* to be true.
  (`:need-for-purpose` *predicate form*) is true if *predicate* is a one-place predicate and an object ϕ such that (`:apply` *predicate* ϕ) is true, is needed for *form* to be true. As an example, (15)a would be true, since `[ENG1]` is needed for the described event to occur. Also, as a general rule, (15)b would be true for any event `?e`, since an engine is needed for any move event to occur.

(15) a.
```
(:need-for-purpose [ENG1] (lf-exists ?E*T-move [E123]
                                     (:ROLE ?E :R-ENGINE [ENG1])
                                     (:OCCURS ?E)))
```

b.
```
(:need-for-purpose (:LAMBDA ?o*T-engine nil) (:OCCURS ?E*T-move))
```

- (`:need-require` *agent obj*) is true if *agent* needs *obj*. Generally this will be true if the object is needed for some purpose (see above).
  (`:need-require` *agent predicate*) is true if *predicate* is a one-place predicate and *agent* needs an object ϕ such that (:apply *predicate* ϕ) is true.

- (`:right-correct` *speechact*) is true if the content of *speechact* is true.

- (`:evaluation` *evaluator object rating*) is true if *evaluator* evaluates *object* as *rating*. Currently only used for *objects* of type `T-plan`

- (`:content` *speechact form*) is true if *speechact* has (propositional) content *form*.

- (`:focus` *speechact obj*) is true if *speechact* has focus *obj*. The focus is used to guide subsequent interpretation and also to guide expectations. An *inform* act can be further classified as an *inform-ref* with the focus as a domain object. A focus of `:yes` or `:no`, on the other hand, will be a *inform-if*.

- (`:interp` *conv-event speechact*) is true if *speechact* is a (partial) interpretation of *conv-event*.

- (`:surf-interp` *conv-event speechact*) is true if *speechact* is the surface interpretation (i.e., the hypothesis generated before act verification) of *conv-event*.

- (`:PREV-CE` *conv-event1 conv-event2*) is true if *conv-event2* is a conversational event which precedes *conv-event1* (within a particular discourse segment).

- (`:NEXT-CE` *conv-event1 conv-event2*) is true if *conv-event1* is a conversational event which precedes *conv-event2* (within a particular discourse segment).

- (`:PREV-ACT` *conv-event speechact*) is true if *speechact* is one of the act interpretations of an event prior to *conv-event* (within a particular discourse segment).

- (`:NEXT-ACT` *conv-event speechact*) is true if *speechact* is one of the act interpretations of an event following *conv-event* (within a particular discourse segment).

6.2.3 KR Operations

The system maintains knowledge using the RHET system [Allen and Miller, 1991] augmented by several Lisp extensions. RHET provides facilities for

- temporal reasoning
- hierarchical belief contexts
- equality and inequality reasoning
- type reasoning

Facilities are provided for asserting, retracting, listing, and querying facts and axioms in belief contexts. Sets of facts can also be moved or copied from one belief context to another. In addition, there are facilities for simplifying complex forms by skolemizing existential expressions, binding definite references to particular objects, and performing lambda application.

6.2.4 Belief Modalities

The belief and proposal modalities described in Section 5.1.1 are implemented as RHET belief contexts [Allen and Miller, 1991]. Figure 6.4 shows the major contexts in the implemented dialogue manager. **T** is the root context, representing general (assumed) mutual beliefs. Other spaces inherit uncontradicted facts from parent spaces, indicated by the arrows. Thus, the **Shared** context, which represents proposals made by one conversant and accepted by the other, will also inherit facts and axioms from **T**, unless these are contradicted by facts directly in **Shared**. **Hprop** represents the (mutually believed) user-proposed space, and **Sprop** the (mutually believed) system-proposed space, each of which inherits from **Shared**. The **Splan** space is where newly elaborated plans from the domain plan reasoner are placed before being conveyed to the user.

There are also a number of transient spaces used in the interpretation process, corresponding to the (private) user and system proposed spaces in Section 5.1.1. Each ungrounded DU will have its own space, inheriting from either **Sprop** or **Hprop**, depending on whether the system or the user is the initiator. Core speech acts which have been recognized during utterances forming `initiate`, `continue`, or `repair` grounding acts will generally add information to the DU space. If and when the DU is grounded (through performance of `ack` acts), the information will be transferred to the appropriate proposal space. Future acts will be interpreted within the context of the current DU space.

In addition, during the act verification phase of speech act interpretation (see Section 6.4.2.2), additional spaces are created to test the speech act hypotheses and to evaluate sequences of acts in a context in which the ramifications of hypotheses about previous acts have been made explicit.

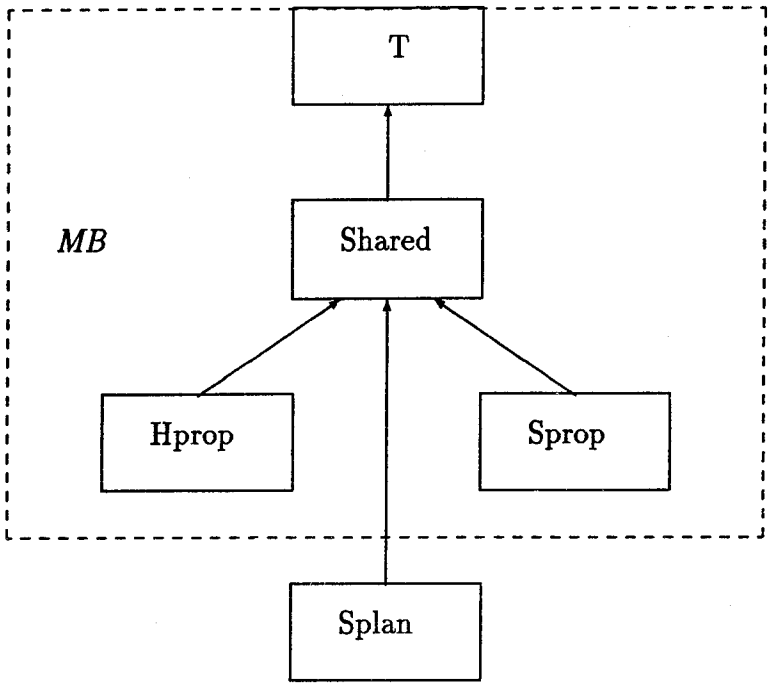

Figure 6.4: TRAINS-93 Permanent Belief Contexts

6.3 Other Aspects of the Discourse Context

6.3.1 Domain Plans

Domain plans are treated as EBTL individuals as discussed in Section 6.2.1. The dialogue manager keeps a stack of active domain plans, ordered by accessibility (although for the dialogues currently handled, only one domain plan is ever used).

Plan recipes are represented as collections of EBTL propositions about a plan. Some examples of plan propositions are shown in (16). (16)a says that the object [MOVE-COMMODITY-5769] is an event in plan [PLAN-5709]. (16)b says that the proposition of this event occurring is a goal of the plan, and (16)c says that the event [LOAD-ORANGES-14935] will achieve a fact in the plan, namely that the commodity of [MOVE-COMMODITY-5769] will be *in* the object that plays the car role of this event at the required time.

(16) a. `(:PLAN-EVENT [MOVE-COMMODITY-5769] [PLAN-5709])`

b. `(:PLAN-GOAL (:OCCURS [MOVE-COMMODITY-5769]) [PLAN-5709])`

c.
```
(:PLAN-ACHIEVES [LOAD-ORANGES-14935]
      (:IN [F-COMMODITY MOVE-COMMODITY-5769] [F-CAR MOVE-COMMODITY-5769]
           [F-PRE2 MOVE-COMMODITY-5769])
      [PLAN-5709])
```

The domain plan reasoner represents recipes as *plan graphs*, in which the facts and events are nodes and the relations between them (e.g., *achieves*, *enables*, *supports*) are

the links. The meanings of these relational predicates and plan graphs are described in more detail in [Ferguson, 1992], and [Ferguson and Allen, 1994] describes the underlying view of plan graphs as *arguments*.

As different belief contexts might contain different sets of these propositions, they will represent different views of the contents of the plan.

6.3.2 Obligations

Obligations are represented in two ways. First, for general obligations, the EBTL predicate `:obligation` is used. This allows for general reasoning about obligations and their consequences within the temporal logic (although in the current system little reasoning is performed about general obligations).

The system also maintains two stacks (one for each conversant) of **discourse obligations** which have not yet been addressed. Each obligation on the stack is represented as a type paired with a content. The stack structure is appropriate because, generally, one must respond to the most recently imposed obligation first. As outlined in Section 5.3, the system will attend to obligations before considering other parts of the discourse context. Most obligations will result in the formation of *intentions* to communicate something back to the user. The current implementation will always wait until it gets the turn before acting on these intentions by uttering something. When the intentions are formed, the obligations are removed from the stack, although they have not yet actually been met. If, for some reason, the system dropped the intention without satisfying it and the obligation were still current, the system would place it back on the stack.

The system currently uses the following obligation types, shown in Table 6.1 with the types of speech acts which generally invoke them[2]. In this table, *self* refers to the obligee and *other* refers to the other agent. Thus an obligation for an agent to `:check-if` results from the performance of a **check** act by the other agent.

obligation type	source of obligation
`:address`	other `request`
`:repair`	disagree with presupposition of other's utterance
`:answer-if`	other `ynq`
`:check-if`	other `check`
`:achieve`	self `accept`

Table 6.1: TRAINS-93 Obligation Types and Sources

The actions which the system will perform to relieve these obligations will be described in Section 6.5.1.

[2]Obligations can also be invoked directly through statements of obligations of the appropriate type.

6.3.3 Communicative Intentions

Communicative intentions are represented in the form of intended speech acts. A list of all intended speech acts is kept as part of the dialogue manager's internal state, and these acts are passed to the **NL Generator** when the system decides to act on its intentions.

Intended speech acts are currently represented as a list of three components:

- the type of the act
- the body of the act
- the reason for performing the act

The type of the act will be one of the grounding or core speech acts, described in Section 6.4.4. The body will be the "propositional content" of the act, and the reason will be the source of the intention. For the current implementation, the reason will be one of `obligation`, `discourse-goal`, or `grounding`. The reasons are meant to help prioritize and orient generation, as well as to provide a facility for backing up the other discourse state if these intentions cannot be realized.

6.3.4 The Turn

Since the TRAINS-93 system works on typed text, input a complete sentence at a time, there's not a whole lot for the turn-taking mechanism to do. Still, the turn is represented as a variable storing the name of the current turn-holder, which can be modified by turn-taking acts (see Section 6.4). Local initiative is not explicitly modelled in the TRAINS-93 implementation.

6.3.5 Grounding Model

A bounded stack of accessible DUs is maintained as described in Section 5.1.6. The current bound is set at 3, meaning that only the top three DUs can be augmented by new grounding acts. Content in older DUs cannot be repaired or acknowledged, although one could reintroduce the material again, and repair the content through new core speech acts indicating a change in beliefs.

Each DU is represented as a structure containing: a variable to represent the initiator (either System or User), another variable representing the state of the DU, a list of the core speech acts which comprise this DU, a list of those acts in the previous list which have already been processed (which means that their contents have been used to update the context beyond the DU), and a belief context which reflects what the proposal space of the initiator would be if this DU were grounded – it is a descendant of the proposal space and includes the effects of the DU's core speech acts.

6.3.6 Discourse Segmentation

The current system does not represent separate discourse segments. It does maintain a list of conversational events (utterances), ordered according to occurrence time. Each conversational event (CE) is associated with the conversation acts which comprise it, and for each core speech act which affects the domain plan, the system stores the resulting plan state and most salient plan graph node for that act. Thus, for argumentation relations which signal a relation to a previous conversational event (e.g., **:so**, **:and-then**), the plan node is available for that previous event. The system currently checks for appropriate nodes from most to least recent conversational events. If discourse segmentation information were kept, this would limit the search to only conversational events in the open segment tree. Argumentation acts would also provide clues as to the segmentation structure – passing up a more recent event by referencing a previous event is a signal that the more recent event is now in a closed sub-segment.

6.4 Conversation Act Implementation

This section presents the implementation in the TRAINS-93 system of the conversation act theory presented in Chapter 4.

6.4.1 Conversation Act Representation

The four types of conversation act described in Chapter 4 are all represented within EBTL. As mentioned in Section 6.2.1.1, core speech acts are represented by a specialization of the structured type `T-action`, `T-speechact`. The other three levels are represented as predications rather than as action individuals. For each utterance, the conversation acts which are determined to have been performed in producing the utterance are stored in a *Conversation Act Frame* which includes slots for a list of acts at each conversation act level.

6.4.1.1 Core Speech Act Representation

Type `T-Speechact` has roles for a speaker and a hearer, as well as a time. In addition, `:Content` and `:Focus` predicates are defined for speech acts. Some of the acts are grouped as *informational speech acts* (subtype `T-Inf-SA`), and *plan-based acts* (subtype `T-Plan-SA`). Figure 6.5 shows the speech act hierarchy.

The core speech acts used by the TRAINS-93 system are listed below.

t-inform Speaker presents Hearer with new knowledge in an attempt to add a new mutual belief. Content is a formula expressing the new information.

t-ynq Speaker asks Hearer to provide information that Speaker is missing but suspects that Hearer may know; imposes a discourse obligation on Hearer to respond. Content is a formula expressing the desired information.

```
Action -- Speechact ------- Inf-SA ------Inform
                                        Supp-Inf
                                        YNQ
                                        WHQ
                                        Check
                            Plan-SA ---- Suggest
                                        Request
                                        Supp-Sug
                            Accept
                            Reject
                            Eval
```

Figure 6.5: Structured Core Speech Act Type Hierarchy

t-check Like a ynq, but Speaker already suspects the answer; Speaker wants to move the proposition in question from individual to mutual belief, or bring the belief into working memory.

t-suggest Speaker proposes a new item as part of a plan. Content is a specification of the suggestion.

t-request Like a suggest, but also imposes a discourse obligation to respond.

t-accept Speaker agrees to a proposal by Hearer; proposal is now shared. Content is a complex type description of the accepted act(s).

t-reject Speaker rejects a proposal by Hearer. Content is a complex type description of the rejected act(s).

t-supp-sug A supplementary suggestion. Like a suggestion, but does not get placed "in focus". The content is a formula, which, if not part of the plan, is added to current plan, and if it cannot be added is responded to immediately, but is otherwise ignored.

t-supp-inf Provides supplementary information. The content is a formula, which, if not known, is added to current knowledge, and if it contradicts current knowledge is responded to immediately, but is otherwise ignored.

t-eval An evaluation by the speaker of the "value" of some (physical or intensional) object.

The contents of domain-plan oriented acts (suggestions, supplementary suggestions and requests having to do with a domain plan) are one of the following.

- (`:goal` *form plan*) The truth of *form* is a goal of *plan*.
- (`:uses` *object form plan*) *object* is used in *plan*, *form* provides background information about how *object* is relevant.

- (:event-in *event-desc plan*) An event of the type described in *event-desc* is part of *plan*

6.4.1.2 Core Speech Act Connectives

The following connectives are used to make up a complex act type from more primitive acts. Each can be used on complex acts, recursively, as well as on basic act types.

- (:or act_1, ..., act_n) occurs iff at least one of act_1, ..., act_n occurs.
- (:and act_1, ..., act_n) occurs iff all of act_1, ..., act_n occur.
- (:ex-or act_1, ..., act_n) occurs iff exactly one of act_1, ..., act_n occurs. The acts act_1, ..., act_n are presumed to be listed in decreasing likelihood of occurrence.
- (:sequence act_1, ..., act_n) occurs iff each of act_1, ..., act_n occurs in a sequence such that each one occurs before the next (i.e., (:TIME-RELN [F-TIME act_1] :B [F-TIME act_2]), ... (:TIME-RELN [F-TIME act_{n-1}] :B [F-TIME act_n])).

6.4.1.3 Argumentation Act Representation

Argumentation acts are represented as predicates which link core speech acts in the proper argumentation relation. We take a fairly pragmatic approach towards the representation and reasoning about argumentation acts. Those relations that are conventionally signaled by surface features (e.g. by clue words such as "so", "no", "okay", purpose clauses) are hypothesized by the surface interpreter (see Section 6.4.2.1) and used by the speech act pruner (see Section 6.4.2.2) to guide further interpretation. In the case of more implicit relationships we often do not identify the precise relation, merely operating on the core speech act level forms. Of course, relations could be identified by analyzing how the content is treated with respect to previous content, but that doesn't seem helpful presently. The argumentation relations implemented in the TRAINS-93 system are listed below.

- (:so act_1 act_2)
  act_1 is relevant to the interpretation of act_2. If act_2 is an informational act, then the truth of its content should be partially supported by act_1. If act_2 is a suggestion, then the suggestion should be about (a part of) the plan dominated by act_1.
- (:and act_1 act_2)
  the interpretation of act_2 is connected to act_1 in some way to form a coherent whole. If act_2 is a suggestion, then it should be part of the same plan as act_1.
- (:and-then act_1 act_2)
  the interpretation of act_2 is connected to act_1 in some way to form a coherent whole. If act_2 is a suggestion, then it should be part of the same plan as act_1. In addition, act_2 should temporally follow act_1.

- (:purpose *act form*)
  act is to be done for the purpose of achieving *form*.

- (:background act_1 act_2)
  act_1 is performed for the purpose of making act_2 more clear to the hearer.

6.4.1.4 Grounding Act Representation

Grounding acts are represented as predicates:
(*Grounding-act-type Actor DU Csa-add-list Csa-delete-list*)

Grounding-act-type is one of: `:init`, `:cont`, `:ack`, `:repair`, `:reqrep`, `:reqack`, or `:cancel`, as developed in Chapter 3 and described in Chapter 4. *DU* is the discourse unit that this act is to be part of (one of the DUs in the grounding memory). *Csa-add-list* is the list of core speech acts which this grounding act adds to the DU. *Csa-delete-list* is the list of core speech acts which are to be removed from the DU by the grounding act (e.g., in the case of a repair).

6.4.1.5 Turn-taking Act Representation

Turn-taking acts are represented as predicates: (*TT-act-type Actor Time*)

TT-act-type is one of the following.

:take-turn the performer takes up the turn and proceeds to speak.

:keep-turn the current turn-holder maintains this status.

:release-turn the turn-holder gives up the turn to another speaker.

Actor is the speaker who performs the turn-taking act. *Time* is the time of occurrence of the act. Currently, times are only distinguished at the level of sentential utterances, so all turn-taking acts which occur as part of the same sentential level utterance will have the same time. The list of turn-taking acts in the conversation act frame for an utterance are assumed to occur in sequential order, so that if a particular utterance has as its act list: `((:KEEP-TURN MANAGER [NOW]) (:RELEASE-TURN MANAGER [NOW]))` this is interpreted as a `keep-turn` followed by a `release-turn`, and the turn would revert to the system after this utterance.

6.4.2 Recognizing Conversation Acts

Core speech acts and argumentation acts are recognized according to the method proposed by Hinkelman, in which surface cues are used to produce a set of speech act *hypotheses*, and then contextual and plan-based information is used to eliminate some of the hypotheses [Hinkelman and Allen, 1989; Hinkelman, 1990]. In the TRAINS system, the **Speech Act Interpreter** produces an underspecified interpretation for an utterance (in the form of a complex act type using the connectives from Section 6.4.1),

and the **Speech Act Pruner** uses contextual and plan-based information to validate or eliminate particular possibilities, binding indexicals to the appropriate values given the current discourse context. The result will be lists of validated core speech acts and argumentation acts, which are added to the conversation act frame for the utterance.

Grounding acts and turn-taking acts are currently recognized in a single-stage process. Since the TRAINS-93 system receives typed rather than spoken input, and language interpretation is performed on a complete sentence rather than incrementally, a complete analysis at these levels is difficult and often superfluous. For example, at the grounding level, there are fewer **continues** and **repairs**. However, in keeping with the theory developed in Chapter 4, and the goal of eventually using the TRAINS system with spoken input, partial interpretations are made which approximate the full theory.

6.4.2.1 Core Speech Act and Argumentation Act Surface Interpreter

As mentioned in Section 6.1.1, the speech act interpreter takes as input the unscoped Logical Form (ULF), which maintains features of the surface input, such as modals and left-clefts. The current speech act interpreter, implemented by Peter Heeman, is a pattern matcher, written as a series of RHET axioms which map features of the ULF input to EBTL speech act templates. These templates are then merged with the output of the scoping and deindexing module, to fill in reference and scoping information, and then translated into EBTL.

Some examples of the kinds of information that are used to build speech act hypotheses are given in Table 6.2. Other argumentation acts are proposed on the basis of cue phrases such as "and", "and then", and "so".

Syntactic Feature	**Resulting Speech Act**
Declarative sentence	`(:ex-or inform check ynq)`
"okay" or "right" or "good"	`accept`
"There is" `?object`	`suggest` use `?object`
Declarative modal (must, should)	`suggest` goal
Interrogative Sentence	`(:ex-or ynq check)`
"right?"	`check` of previous speech act
Imperative sentence	`request`
Relative clause	`supp-inf` & `:background` argumentation act
Purpose clause	`supp-sug` & `:purpose` argumentation act

Table 6.2: Surface Speech Act Rules

The speech act interpreter uses the EBTL quantifiers and special predicates to represent indexical arguments that the pruner can fill in from the context of interpretation. For example, a suggested action is taken to mean that the action forms a part of a plan[3]. Since the surface interpreter does not use the context or have the ability to determine whether a particular action might form a part of a given plan, it will use a

[3]Although this might be a trivial plan to just achieve the performance of a suggested action.

quantified variable to refer to the plan argument of the content of the suggestion. If the action is to be part of a plan that is already known about or the surface form reveals some information about the plan, then a definite reference will be used. Otherwise, an indefinite quantifier will be used. For example, (17)a says that there is some plan, the current plan, which has [event] as a component event (although the speech act interpreter does not know what the current-plan *is*, and this is left for the pruner to determine). (17)b, on the other hand, merely says there is some plan which has this constituent event. Again, the pruner will determine whether the mentioned plan has any relation to any known plans, or if it introduces a new plan.

```
(17) a. (:THE ?P*T-PLAN ?DM*T-ANYTHING (:CURRENT-PLAN ?P*T-PLAN )
            (:EVENT-IN (:OCCURS [event]) ?P*T-PLAN))

     b. (:LF-EXISTS ?P*T-PLAN ?DM*T-ANYTHING NIL (:EVENT-IN (:OCCURS [event]) ?P*T-PLAN))
```

Similarly, the cue word "so" in initial or solitary position, is taken (under one interpretation) to be a signal of an argumentation act which relates a previous conversational event to a succeeding conversational event in some sort of summary or evidentiary manner. However, the surface interpreter does not know *which* previous event is referred to, and, if called on just the particle itself, might not even know what is to follow.

At the surface interpretation level, this can be formulated elegantly as (18), where [CE] is the conversational event of uttering "so". It is up to the pruner to find speech acts PA and NA such that PA is a previous event (in the same discourse segment tree) and NA is a following event, such that the :so relationship holds over those two events.

```
(18) (:THE ?PA*T-SPEECHACT ?P*DM (:PREV-CE [CE] ?PA*T-SPEECHACT)
         (:THE ?NA*T-SPEECHACT ?N*DM (:NEXT-CE [CE] ?NA*T-SPEECHACT)
             (:SO ?PA*T-SPEECHACT ?NA*T-SPEECHACT)))
```

6.4.2.2 Core Speech Act and Argumentation Act Pruning

The speech act pruner will evaluate the hypotheses created by the surface interpreter in the current context, deciding which acts have actually occurred, and filling in indexical information for those acts.

The basic principle behind the core speech act pruning is to test the applicability conditions of each act, eliminating those that are not coherent with the current context. These conditions are described below. If the speech act connectives (:or, :and, and :xor) are part of the hypotheses, then the results are built up from the individual applicable acts as follows:

(:or act_1 ... act_n) all act_i which are applicable have occurred

(:and act_1 ... act_n) if all act_i are applicable, they all have occurred, otherwise none have occurred.

(:xor act_1 ... act_n) only the first act_i which is applicable has occurred

Sequences of speech acts are more difficult to assess, since each sequential act must be evaluated in an environment which has already been updated by the preceding acts. For each act in the sequence, a temporary belief context is created, and the effects of the previous acts are added to this context, just as they would be to the main contexts if they are finally judged to have happened, as outlined in Section 6.4.3.1.

Argumentation acts must generally be evaluated in concert with core speech acts. Because the argumentation acts used here relate core speech acts together with other material, it is necessary to know which core speech acts occur in order to determine if the argumentation acts occur. On the other hand, since argumentation acts specify how the contents of core speech acts cohere with other material, it is important to use this information in deciding whether a core speech act occurs, and what its effects are. This is particularly true for domain plan suggestions in which the argumentation act will help specify how a proposal relates to known elements of the current plan.

The applicability of individual core speech acts is determined as follows. In the description below, `context` refers to the belief context in which the act is evaluated. This could be the current *proposal context* of the speaker (**Sprop or Hprop**, as described in Section 6.2.4), or a currently open DU context (see Section 6.3.5), or a transient child of one of these created to evaluate a sequence of speech act hypotheses, as described above.

t-inform An `inform` act is applicable only if the content of the proposed act is not already mutually believed. The pruner thus tries to prove the content of the `inform` in the **Shared** context, rejecting the hypothesis that the act occurred if the content can be proved. In addition, for potential informs without a specified propositional content (e.g., a plain "yes" or "no"), the discourse obligation stack (described in Section 6.3.2) for the speaker is consulted to see if there is an appropriate proposition which the speaker might be asserting.

t-ynq A `ynq` is applicable only if the polarity of the content of the proposed act is not already known by the speaker. The pruner tries to prove the content of the `ynq` in the speaker proposed space (**Hprop** for the user or **Sprop** for system), rejecting the hypothesis that the act occurred if the content can be proved.

t-check A check should be applicable either in the case in which the speaker has a weak belief (or *suspicion* [Bunt, 1989]) that the content holds, but is not convinced that this belief is mutual. Since we have only a binary belief model and cannot represent degrees of belief, we do not rule out the applicability of a `check` act.

t-supp-inf A supplementary inform is never deemed inapplicable, although it may be considered erroneous (see Section 6.4.3.1).

t-suggest A suggestion is applicable if the system can determine how the suggestion can cohere with the current plans. Currently only domain suggestions are allowed, and coherence is determined by calling the domain plan reasoner to `incorporate` the content into a domain plan. If the domain plan reasoner decides that the

incorporation is okay and the suggestion adds new content to the plan, the suggestion is deemed applicable. Argumentation acts will often play a key role in determining the coherence of suggestions, since they help constrain or focus the domain planner's search.

t-request same as for `t-suggest`, but conversation level requests to perform speech acts are also allowed.

t-supp-sug A supplementary suggestion is never deemed inapplicable, although it may be considered erroneous (see Section 6.4.3.1).

t-accept An `accept` is applicable if there are any speech acts performed by the other agent which have not been accepted but are accessible.

t-eval An `eval` act is applicable if its content (specifying the object that is evaluated and its value) is consistent with the knowledge in the speaker proposed belief context.

6.4.2.3 Domain Plan Incorporation and Argumentation Acts

Suggestions, requests, and supplementary suggestions of augmentations to a domain plan are evaluated by calling the domain plan reasoner to *incorporate* the suggestion into the plan. This incorporation involves a plan recognition process of inferring the necessary additions and assumptions so that the new item forms a coherent part of the plan, supporting the goals of the plan. If the incorporation is successful, a list of plan propositions is returned which comprise the additions to the plan.

Argumentation acts may affect the incorporate call by providing extra information which can be used to help focus the search. Different hypotheses about argumentation acts are verified by making prioritized serial calls to the domain plan reasoner until one of them returns successfully. In this case, the argumentation acts which contributed to the successful call are considered to have been performed.

Each *incorporate* call includes an EBTL formula to be incorporated and a domain plan term representing the plan which should be augmented. Additional information, including that from argumentation acts are passed in as values to keyword parameters. The keywords, their meanings, and the features which lead to them are:

:incorp-type The type of formula to be incorporated. Either `:goal`, `:event`, or `:role-filler`.

:focus An object that the incorporation is focused on. For an `:incorp-type` of `:role-filler`, this object will fill the role.

:bg Background information which relates to the formula to be incorporated. This can come either from a `:BACKGROUND` argumentation act, or, in a case in which a declarative utterance about the domain is used to suggest the use of a focused object, the content of the declarative utterance will be the background. For an

utterance such as "There are oranges at Corning.", which might be a suggestion to use the focused collection of oranges, the fact that the oranges are at Corning will be given as the background argument.

:context The root of a subgraph of the plan which constrains the search. If this argument is provided with an `:incorp-type` of `:goal` then the proposition to be incorporated will be a sub-goal of the plan. The `:context` argument comes from a `:so` argumentation act, with a meaning that the new formula to be incorporated will support in some way (e.g., *generation*, *enablement*, etc.) the occurrence or performance of the context.

:purpose Like `:context`, except that the `:purpose` is not assumed to be already part of the plan. Whereas the `:context` argument will be an item that is already part of the plan, the `:purpose` must first itself be incorporated before the main formula can be incorporated to support it. This argument will come from a `:PURPOSE` argumentation act.

:temp-seq Performance of the event mentioned by this argument is assumed to (immediately) precede the performance of the event in the formula to be incorporated. This argument comes from an `:and-then` argumentation act.

The details of the incorporation algorithm can be found in [Ferguson, 1994], while the underlying view of plans as *arguments* is presented in [Ferguson and Allen, 1994].

6.4.2.4 Recognizing Grounding Acts

The sentential-level utterance processing by the TRAINS-93 system trivializes the recognition of grounding acts to some degree – there are fewer utterances in general, and particularly fewer `continues` and `repairs`. Still, the same ideas can be used to recognize grounding acts in the sentential input.

An utterance containing core speech act interpretations which is interpreted when there are no ungrounded accessible DUs will be an `:init`.

Other act interpretations depend on the comparison of the new core speech acts to the CSA's in the accessible DUs. If the new utterance contradicts or (potentially) changes the previous material, it is seen as a `:repair`. If it does not, and the speaker is the same as the *initiator* of the DU, then it is a `:cont`. A full implementation would also determine how closely related the new utterance is to the previous ones (e.g., by checking argumentation acts which include both) and recognizing an `:init` of a new DU in the case in which they are not sufficiently related.

If the speaker is the responder and the utterance contains new CSA's or argumentation acts that refer to the old acts, such as an `accept` or `reject`, then this is seen as `:ack`. If there are also other acts containing new material, the utterance will also contain an `:init` of a new DU.

`:reqAck`, `:reqRep`, and `:cancel` are assumed to require prosodic or lexical cues, generally within a sentence, and are not recognized by the current implementation.

6.4.2.5 Recognizing Turn-Taking Acts

The turn-taking facility in a typed-input, sentential-level utterance system is especially primitive as most of the cues that are used for turn-taking (e.g., intonation and pauses) are not present. We include a degenerate turn-taking act recognition to allow the interaction with the rest of the system to proceed according to the spoken language model presented in the previous chapters. In order to model the typed input as spoken language, turn-taking action is approximated as two turn-taking acts per utterance, one for the action performed by beginning an utterance, (a `:keep-turn` or `:take-turn`), followed by one for the action performed by ending the utterance (a `:keep-turn` or `:release-turn`). In addition, a `:release-turn` is assumed to give the turn to the other conversant (just as `assign-turn` in Chapter 4). A full turn-taking analysis would include extra `:keep-turn` acts within the utterance, and, possibly `:release-turn` acts followed by `:take-turn` acts if the speaker released the turn but took it back again without intervention from the other speaker.

For the utterance-initial act, if the speaker already had the turn, this is a `:keep-turn` act. If the speaker did not have the turn then it is a `:take-turn` act. There are several strategies for determining the sentence final act. First, if the speaker is still "speaking" when the utterance has been processed (there is more input waiting on the input stream for the speaker), then this is a `:keep-turn` act. Otherwise, a `:release-turn` act is assumed.

In addition, the interface provides explicit extra-linguistic signals for keeping and releasing the turn. If "`<rt>`" is typed anywhere in the input utterance, this will be treated as an explicit signal of releasing the turn to the system. If "`<kt>`" is typed, then the turn will remain with the user, even if the following input is not ready when the utterance has been processed. This facility allows one to manipulate the turn-taking pattern explicitly, so that particular interaction patterns can be tested.

6.4.3 Effects of Conversation Acts

This section describes the effects of performance of conversation acts on the discourse context.

6.4.3.1 Effects of Core Speech Acts

t-inform Performance of an `inform` will add the content of the inform to the speaker proposed space (**Hprop** for the user or **Sprop** for system).

t-ynq Performance of a `ynq` will add a *discourse obligation* for the hearer to `:answer-if` the content of the `ynq`.

t-check Performance of a `check` will add a *discourse obligation* for the hearer to `:check-if` the content of the `check`.

t-supp-inf Performance of a supplementary inform will have different effects depending on the system's beliefs about the content. If the content is contradicted by

knowledge in the speaker proposed space, an obligation for the system to repair the utterance is added. Otherwise, the fact is added to the speaker proposed space (although it might already be there).

t-suggest The effects of a suggestion will be to place in the speaker proposed space the plan propositions (See Section 6.3.1) which result from the incorporation of the suggestion (See Section 6.4.2.3).

t-request A request will have the same effects as a suggestion, but will additionally add a discourse obligation for the hearer to **address** the request.

t-supp-sug A supplementary suggestion will have the same effects as a suggestion, but if the content can not be incorporated, a discourse obligation to repair is incurred.

t-accept Performance of an **accept** will cause the information added by the accepted acts to be moved from the speaker proposed space to the shared space. E.g., An **inform** by the user will cause information to be added to the **Hprop** space. An **accept** of this **inform** by the system will cause this information to be moved to **Shared**.

t-eval Performance of an **eval** will add the evaluation to the speaker proposed space (**Hprop** for the user or **Sprop** for the system).

6.4.3.2 Effects of Argumentation Acts

The main effects of argumentation acts are in providing clues to the correct meaning of the core speech acts that they relate. The effects will therefore show up in the effects of the core speech acts.

6.4.3.3 Effects of Grounding Acts

Grounding acts will have two types of effects. First, they will update the state of the DU of which they are a part, as described in Table 3.1. Additionally, they will cause the belief structures to be updated by adding the effects of the core speech acts which are associated with the grounding act to the belief context associated with the DU. The acts will also have the following individual effects.

:init Create a new DU and push to the front of the DU stack, perhaps rendering an older DU inaccessible.

:ack The previously unprocessed acts from the DU now have their full effects. This includes moving information associated with the acts from the belief context to the speaker proposed space, as well as (for **accepts**) moving information to **Shared**, and adding (for questions or **requests**) or removing (for answers) discourse obligations.

:repair The replaced speech acts will be removed from the DU, and their effects will be removed from the DU belief context.

:reqRep In addition to potentially removing some speech acts and/or effects from the DU, a discourse obligation will be added for the hearer to perform a repair.

:reqAck A discourse obligation will be added for the hearer to perform an acknowledgement.

6.4.3.4 Effects of Turn-taking Acts

The effects of turn-taking acts are very simple: the **turn** is updated, according to Table 6.3.

tt-act	Turn holder after [time]
(:TAKE-TURN ?self [time])	?self
(:KEEP-TURN ?self [time])	?self
(:RELEASE-TURN ?self [time])	?other

Table 6.3: Turn-taking Act Effects

6.4.4 Intended Speech Acts

This section describes the acts which the system intends to produce. Currently the system always intends to release the turn after each utterance, and no provisions are made for producing argumentation acts. The general form of the intentions (*act-type body reason*) is described above in Section 6.3.3.

6.4.4.1 Core Speech Act Types

:inform *Body* is a formula expressing the new information.

:inform-if A specialization of inform, where one is asserting that a contextually relevant fact has a particular truth value. *Body* will be a focus frame with a focus indicating the polarity.

:inform-ref A specialization of inform, where one is asserting that a contextually relevant property holds of an object. *Body* will be a focus frame with a focus indicating the object.

:ynq *Body* is a formula describing the desired information.

:suggest *Body* is a specification of the suggestion.

:request *Body* is a specification of the request.

:accept *Body* is a list of the accepted act(s).

:reject *Body* is a list of the rejected act(s).

:eval *Body* is the evaluation of an object.

6.4.4.2 Grounding Act types

:ack *Body* is a list of the acknowledged act(s).

:repair Speaker attempts to repair a previous utterance. Body is a pair of formulae, the first describing the material to be deleted and the second describing material to be added.

:req-repair Speaker requests hearer to repair a previous utterance. Body is a formula describing the desired repair.

6.5 Discourse Actor Actions

This section describes the specific actions performed by the implemented discourse actor in more detail than was provided in Section 5.3.

6.5.1 Actions based on Obligations

The action taken depends on the type of obligation and other features of the discourse context. Each obligation type from Table 6.1 is discussed below:

:address The system addresses requests first by deciding whether or not to accept them. The only types of requests the system currently knows how to deal with are requests to adopt proposals about new contents in a domain plan, and to evaluate a plan. The system currently accepts all requests. For the former case, the system simply adds an intention to accept the request. In the latter case, the system must first check if the plan is already known to be complete. If not, it will call the domain plan reasoner to elaborate the plan. The system then adds an intention to perform an **eval** speech act. If it is decided that the plan is already complete, this evaluation will indicate no problem with the plan. Otherwise it will explain that there are missing features (which will also be suggested to the user).

:repair The system will add an intention to perform a `:repair` grounding act. It will also immediately call the generator to act on this intention, even if it does not yet have the turn.

:answer-if First, the system will check to see if it believes the body of the obligation. If so, the system adopts an intention to **inform** that it is true. If not, the system checks to see if it believes the body false. If so, it will **inform** that the body is false. If neither belief holds, then the system will decide if a decision is warranted. For example, if the user is asking whether the system accepts a speech act and the system has not yet decided whether to accept it or not, it will not be very informative to just answer that it doesn't know. Instead the system will make the decision now and then add an intention to report on it (currently the system always accepts suggestions). If the question is just about the facts of the domain, then, if the system doesn't know, it will be appropriate to indicate that.

:check-if The system will first check if the body of the obligation holds according to the system's private beliefs (belief context `*splan*`). If so, the system adopts an intention to indicate agreement. If the body is believed to be false, the system will adopt an intention to indicate disagreement. Otherwise it will indicate that it doesn't know.

:achieve Currently, the only discourse-level action which the system reasons about achieving are those of type `T-Find-Plan`. The system deals with an obligation to achieve this action by, first, adding the top-level goal to have a plan-conversation if it is not already there. Next, if one does not already exist, a new **current domain plan** object is created. Then, the **event** role of the find-plan action is instantiated as the **goal** of the plan, and a call is made to the domain plan reasoner to *incorporate* this goal into the current domain plan. Domain-level actions are achieved by placing them in the plan and then executing it.

6.5.2 Other Actions

The other actions of the discourse actor are, for the most part, as described in Section 5.3.

6.6 Annotated Trace of a Sample Conversation

As an example of the system in operation, we show the operation of the dialogue manager and conversation act analysis on a simplification of the collected dialogue used in Section 4.3 (Dialogue 91-6.1 from [Gross *et al.*, 1993]). Figure 6.6 shows the abridged dialogue in full. The utterance numbering system used here reflects the relation to the turn and utterance numbering used in the original. '3-7' represents utterance 7 within turn 3. '=' is used to indicate merged utterances. Thus '3-3=6' spans four utterances in turn 3 of the original, and 9=13 replaces turns 9 through 13 in the original.

The system starts with no discourse goals or any conversational state, but with background knowledge corresponding to the map in Figure 6.7.

The following EBTL constants (see Section 6.2.1) relevant to processing the dialogue are defined in the initial system knowledge, with their types given in parentheses:

[HUM] (T-Human) The user

[SYS] (T-Human) The system

[SYSHUM] (T-Human) The system and user together acting as a joint agent

[AVON] [BATH] [CORNING] [DANSVILLE] (T-City) Cities in the TRAINS world

[ORANGES-1] (T-Oranges) The oranges at Corning

[BOXCAR-1] (T-Boxcar) The boxcar at Dansville

Utt. #	Speaker: Utterance
1	U: We better ship a boxcar of oranges to Bath by 8 AM.
2	S: Okay.
3-3=6	U: So we need to get a boxcar to Corning, where there are oranges.
3-7	U: There are oranges at Corning.
3-8	U: Right?
4	S: Right.
5-1	U: So we need an engine to move the boxcar.
5-2	U: Right?
6	S: Right.
7-1=2	U: So there's an engine at Avon.
7-3	U: Right?
8	S: Right.
9=13	U: So we should move the engine at Avon, engine E1, to Dansville to pick up the boxcar there.
14	S: Okay.
15-2=4	U: And move it from Dansville to Corning.
15-5=7	U: Load up some oranges into the boxcar.
15-8=10	U: And then move it on to Bath.
16	S: Okay.
17	U: How does that sound?
18-3	S: That's no problem.
19	U: Good.

Figure 6.6: Sample dialogue processed by TRAINS-93.

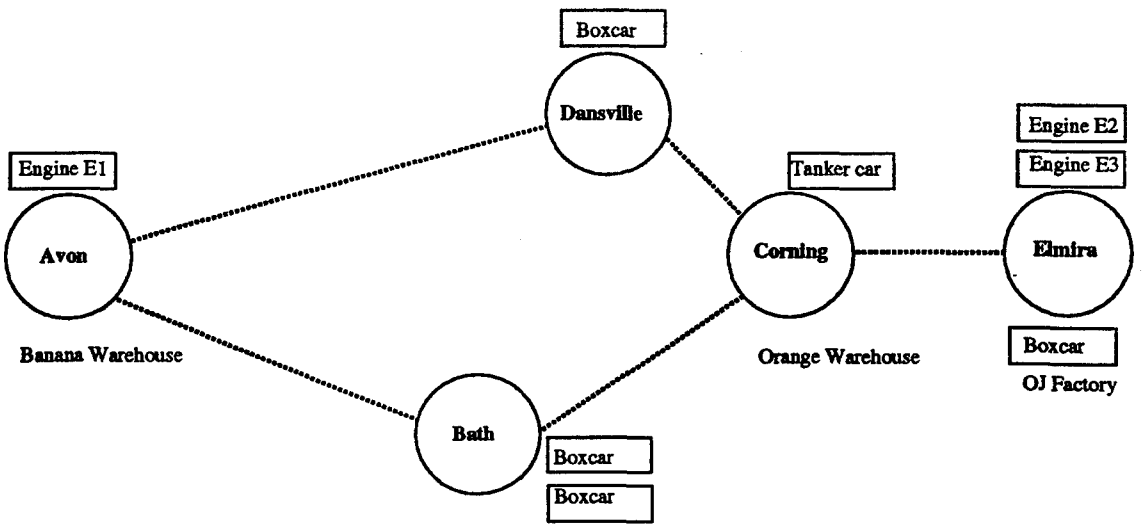

Figure 6.7: Trains World Set-up for Example Conversation

[ENGINE-1] (T-Engine) The engine at Avon

[WAREHOUSE-1] (T-Warehouse) the orange warehouse at Corning

[NOW] (T-Time) the current time interval of the conversation, spanning the times of the individual utterances. Around midnight in real time.

In addition, the following Lisp constants are used in some aspects of the representation:

SYSTEM The System

MANAGER The User

6.6.1 Starting the Conversation

The system first prints a welcoming message and waits for the user to start the dialogue:

SYSTEM: Hello, this is the TRAINS system.
I can help you plan and execute actions in the TRAINS World.

The initial state of the discourse context is shown in Figure 6.8.

Discourse Obligations:
Turn Holder: Manager
Intended Speech Acts:
DU Stack:
Unaccepted Proposals:
Discourse Goals: `have-plan-conversation`
Domain Plan Stack:

Figure 6.8: Discourse Context before Utterance 1

6.6.2 Interpreting utterance 1

USER: we better ship a boxcar of oranges to Bath by eight a.m.

The event of this utterance is named `[CE10]`, and the core speech act hypotheses for the interpretation of this utterance are shown in (19), where the individual speech acts are defined as follows: The `SPEAKER` and `HEARER` and `TIME` roles for each of these acts are, respectively, `[HUM]`, `[SYS]`, and `[NOW1]`, the time of utterance of `[CE10]`.

(19)
```
(:SURF-INTERP [CE10]
        (:OR (:EX-OR [ST-INFORM-0001] [ST-CHECK-0002] [ST-YNQ-0003])
             [ST-SUGGEST-0004]))
```

Additionally, `[ST-INFORM-0001]`, `[ST-CHECK-0002]`, and `[ST-YNQ-0003]` all have the same content, namely that shown in (20).

```
(20) (:OBLIGATION [SYSHUM]
       (:LF-EXISTS ?v4438*T-ORANGES [X3]
           (:QUANTITY ?v4438*T-ORANGES (:BOXCAR-UNIT 1))
           (:LF-EXISTS ?L*T-MOVE-COMMODITY [E0]
               (:APPLY
                 (:LAMBDA ?E*T-MOVE-COMMODITY
                     (:AND (:ROLE ?E*T-MOVE-COMMODITY :R-AGENT [SYSHUM])
                           (:ROLE ?E*T-MOVE-COMMODITY :R-COMMODITY ?v4438*T-ORANGES)
                           (:ROLE ?E*T-MOVE-COMMODITY :R-DST [BATH])
                           (:TIME-RELN [F-TIME ?E*T-MOVE-COMMODITY] :B [8AM])))
                 ?L*T-MOVE-COMMODITY)
                 (:OCCURS ?L*T-MOVE-COMMODITY)))
       [E7])
```

This is the translation of the surface logical form of the utterance into EBTL, namely that the user and system have an obligation that some collection of oranges of size one boxcar load (with a discourse marker [X3], which could be used in future interpretations to refer to the same group) be moved to Bath before eight a.m. The time of the obligation is [E7], which is set by the Scope and Deindexing module to be a sub-situation of the current situation of evaluation.

[ST-SUGGEST-0004] has the content given in (21). This means that there is some plan which has as its goal that a move-commodity event occurs and has the features specified above.

```
(21) (:LF-EXISTS ?P*T-PLAN ?DM*T-ANYTHING NIL
       (:GOAL
        (:LF-EXISTS ?v4574*T-ORANGES [X3]
            (:QUANTITY ?v4574*T-ORANGES (:BOXCAR-UNIT 1))
            (:LF-EXISTS ?L*T-MOVE-COMMODITY [E0]
                (:APPLY
                  (:LAMBDA ?E*T-MOVE-COMMODITY
                      (:AND (:ROLE ?E*T-MOVE-COMMODITY :R-AGENT [SYSHUM])
                            (:ROLE ?E*T-MOVE-COMMODITY :R-COMMODITY ?v4574*T-ORANGES)
                            (:ROLE ?E*T-MOVE-COMMODITY :R-DST [BATH])
                            (:TIME-RELN [F-TIME ?E*T-MOVE-COMMODITY] :B [8AM])))
                  ?L*T-MOVE-COMMODITY)
                 (:OCCURS ?L*T-MOVE-COMMODITY)))
        ?P*T-PLAN))
```

Thus the entire complex hypothesis for this utterance is that either exactly one of the informational acts (inform, check, or ynq) was performed, or the suggestion was performed (or both).

The speech act pruner will now try to evaluate the hypotheses. For [ST-INFORM-0001], this will involve trying to prove the content (given in (20)) in the **Shared** context. This proof fails (since the system had no prior knowledge of this obligation), and thus the inform possibility is applicable. Given the **:ex-or** construct, the results of evaluating [ST-CHECK-0002] and [ST-YNQ-0003] are unimportant.

Since [ST-SUGGEST-0004] is a domain suggestion, evaluating it will involve a call to the domain plan reasoner to *incorporate* the content of the suggestion. First, the

dialogue manager will attempt to determine the plan argument of the content. To do this, it checks the **domain plan stack** to see if there is already a current plan that is under discussion. In this case there is none, so a plan term, [PLAN-4719], is created and placed on the top of the stack. Initially, nothing is known about this plan. However the results of the incorporate call, (22), may provide some information.

(22)
```
(INCORPORATE
   (:LF-EXISTS ?v4574*T-ORANGES [X3]
      (:QUANTITY ?v4574*T-ORANGES (:BOXCAR-UNIT 1))
      (:LF-EXISTS ?L*T-MOVE-COMMODITY [E0]
         (:APPLY
           (:LAMBDA ?E*T-MOVE-COMMODITY
              (:AND (:ROLE ?E*T-MOVE-COMMODITY :R-AGENT [SYSHUM])
                    (:ROLE ?E*T-MOVE-COMMODITY :R-COMMODITY ?v4574*T-ORANGES)
                    (:ROLE ?E*T-MOVE-COMMODITY :R-DST [BATH])
                    (:TIME-RELN [F-TIME ?E*T-MOVE-COMMODITY] :B [8AM])))
            ?L*T-MOVE-COMMODITY)
         (:OCCURS ?L*T-MOVE-COMMODITY)))
   [PLAN-4719]
   NIL
   :INCORP-TYPE :GOAL)
```

The call returns successfully, giving the plan propositions shown in (23) as the results of the incorporation. This means that the occurrence of the designated move-commodity event with the features described above is the goal of the plan. These propositions are stored with [ST-SUGGEST-0004] as its effects.

(23)
```
(:PLAN-GOAL (:OCCURS [MOVE-COMMODITY-4779]) [PLAN-4719])
(:PLAN-EVENT [MOVE-COMMODITY-4779] [PLAN-4719])
(:PLAN-PREMISE (:OCCURS [MOVE-COMMODITY-4779]) [PLAN-4719])
```

The pruner now decides, since both [ST-INFORM-0001] and [ST-SUGGEST-0004] are applicable, that both have in fact been performed.

For grounding act analysis, since there are no accessible DUs, this utterance must be the initiation of a new DU, named DU-1, with the user as the *Initiator*, and a CSA-List including [ST-INFORM-0001] and [ST-SUGGEST-0004]. The effects of these acts are now added to the belief context for this DU, #<context SB-G4940>, a child of **Hprop**. This involves asserting both (20), for the inform, as well as each of the propositions in (23) for the suggest.

Turn-taking analysis comes up with the acts shown in (24), since the user was assumed to have the turn when the utterance was performed, and there was nothing to indicate keeping the turn at the conclusion of the utterance.

(24)
```
(KEEP-TURN MANAGER [NOW1]) (RELEASE-TURN MANAGER [NOW1])
```

The final set of performed conversation acts is shown in (25), while Figure 6.9 shows the relevant parts of the discourse state after interpretation of this utterance. In this

and subsequent discourse context figures, we show only the relevant parts of the context, eliminating those features that remain unchanged and have no effect on the deliberation.

(25) :CSAS [ST-INFORM-0001] [ST-SUGGEST-0004] :ARGS NIL
:GAS (:INIT MANAGER DU-1 ([ST-INFORM-0001] [ST-SUGGEST-0004]) NIL)
:TTAS (KEEP-TURN MANAGER [NOW1]) (RELEASE-TURN MANAGER [NOW1])

Turn Holder: System
DU Stack:
DU-1 :INITIATOR MANAGER :STATE 1 :CONTEXT #<context SB-G4940>
:CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004]) :PROCESSED NIL
Domain Plan Stack: [PLAN-4719]

Figure 6.9: Discourse Context after Utterance 1

6.6.3 Acting after the first utterance

Following the algorithm in Section 5.3: since there are no outstanding discourse obligations, the system has the turn, and there are no intended speech acts, the system addresses the grounding situation. Since the top DU is initiated by the user and is in State 1, the system decides to acknowledge the CSA's in this DU ([ST-INFORM-0001] and [ST-SUGGEST-0004]). This will result in an :ack intended speech act. Additionally, the system will update its internal structure to include the effects of acknowledging the acts. In this case, that means asserting (20) (for the inform) and (23) (for the suggest) to the speaker proposed space – **Hprop**. The resulting discourse context will be as shown in Figure 6.10.

Turn Holder: System
Intended Speech Acts: (:ACK ([ST-INFORM-0001] [ST-SUGGEST-0004]) GROUNDING)
DU Stack:
DU-1 :INITIATOR MANAGER :STATE 1 :CONTEXT #<context SB-G4940>
:CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004])
:PROCESSED ([ST-INFORM-0001] [ST-SUGGEST-0004])
Unaccepted Proposals: ([ST-INFORM-0001] [ST-SUGGEST-0004])

Figure 6.10: Discourse Context after deciding to ack first Utterance

Since acknowledgements can often "piggyback" on other utterances, the system does not immediately produce the acknowledgment, but continues to deliberate to find other things to say. After the acts from utterance 1 have been processed, the system considers the unaccepted proposals, and decides to accept them both. This leads to the discourse state in Figure 6.11.

Finally, because of the intentions, the NL generator will be called to produce an utterance satisfying the intentions. This will be utterance 2 of the dialogue:
SYSTEM: okay.

```
Turn Holder: System
Intended Speech Acts: (:ACK ([ST-INFORM-0001] [ST-SUGGEST-0004]))
                      (:ACCEPT [ST-INFORM-0001]) (:ACCEPT [ST-SUGGEST-0004])
DU Stack:
 DU-1  :INITIATOR MANAGER :STATE 1        :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004])
       :PROCESSED ([ST-INFORM-0001] [ST-SUGGEST-0004])
Unaccepted Proposals: ([ST-INFORM-0001] [ST-SUGGEST-0004])
```

Figure 6.11: Discourse Context after deciding to accept first Utterance

6.6.4 Interpreting utterance 2

Even though the system knows what it was *trying* to say, the utterance is fed back through the language interpretation modules, as shown in Figure 6.1[4]. This utterance event is named [CE187], and the only core speech act hypothesized is [ST-ACCEPT-0005], the specifics of which are given in (26).

(26)
```
(:AND (:ROLE [ST-ACCEPT-0005] :R-SPEAKER [SYS])
      (:ROLE [ST-ACCEPT-0005] :R-HEARER [HUM])
      (:ROLE [ST-ACCEPT-0005] :R-TIME [F-TIME [CE187]])
      (:CONTENT [ST-ACCEPT-0005]
                (:LAMBDA ?SA*T-SPEECHACT
                     (:AND (:ROLE ?SA*T-SPEECHACT :R-SPEAKER [HUM])
                           (:ROLE ?SA*T-SPEECHACT :R-HEARER [SYS])
                           (:OCCURS ?SA*T-SPEECHACT)
                           (:UNACCEPTED ?SA*T-SPEECHACT)))))
```

The content specifies that this act could accept any speech act(s) for which the speaker role was the user, the Hearer role was the system, and which occurred and has not yet been accepted. To evaluate the applicability of this act hypothesis, the system looks for any speech acts which meet the criteria specified in the content. Both [ST-INFORM-0001] and [ST-SUGGEST-0004] meet the criteria, and so the pruner accepts this hypothesis, stipulating further that [ST-ACCEPT-0005] accepts both of these speech acts.

Grounding act analysis treats this as an :ack by the system of the top DU, DU-1, adding [ST-ACCEPT-0005] to the CSA-list for this DU. Updating on the basis of this :ack will change the state of the DU from 1 to F, as well as updating on the basis of the acts in the CSA-list which are not in the list of processed acts, in this case, just [ST-ACCEPT-0005]. Updating on the accept will cause the effects of the accepted acts to be transferred from the speaker proposed belief context (in this case, **Hprop**) to the **Shared** context. This means that the items in (20) (for [ST-INFORM-0001]) and (23) (for [ST-SUGGEST-0004]) are moved from **Hprop** to **Shared**.

[4]This allows the system to check how the utterance will be interpreted by the user given the current context. If this diverges from the intended meaning, the system should intend a **repair**. Feeding the utterance through the language interpretation modules also allows the modules to update their own context on the basis of what has actually happened.

In addition the turn-taking acts in (27) are produced, where [NOW2] is the time of utterance of [CE187].

(27) (KEEP-TURN SYSTEM [NOW2]) (RELEASE-TURN SYSTEM [NOW2])

The conversation acts performed are shown in (28)

(28) :CSAS [ST-ACCEPT-0005] :ARGS NIL
:GAS (:ACK SYSTEM DU-1 ([ST-ACCEPT-0005]) NIL)
:TTAS (KEEP-TURN SYSTEM [NOW2]) (RELEASE-TURN SYSTEM [NOW2])

Figure 6.12 shows the updated discourse context. Since there are no obligations and the user now has the turn, the system waits for a user utterance.

Turn Holder: Manager
DU Stack:
DU-1 :INITIATOR MANAGER :STATE F :CONTEXT #<context SB-G4940>
:CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
:PROCESSED ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])

Domain Plan Stack: [PLAN-4719]

Figure 6.12: Discourse Context after Utterance 2

6.6.5 Interpreting utterance 3-3=6

USER: so we need to get a boxcar to Corning, where there are oranges. <kt>

This is interpreted as a sequence of two conversational events, [CE249], which covers the "so" discourse marker, and [CE251] which includes the sentence that follows. The only core speech act possibility for [CE249] is [ST-ACCEPT-0006], with content given in (29).

(29) (:LAMBDA ?SA*T-SPEECHACT
(:AND (:ROLE ?SA*T-SPEECHACT :R-SPEAKER [SYS])
(:ROLE ?SA*T-SPEECHACT :R-HEARER [HUM])
(:OCCURS ?SA*T-SPEECHACT)
(:UNACCEPTED ?SA*T-SPEECHACT)))

For [CE251], the relative clause leads to a supplementary inform, [ST-SUPP-INF-0011], with the content given in (30)

(30) (:LF-EXISTS ?v6247*T-ORANGES [X240] NIL (:AT ?v6247*T-ORANGES [CORNING] [E242]))

The main clause leads to hypotheses similar to that for utterance 1: alternatives of an inform, check, or ynq of the content given in (31), combined with the suggestion with content given in (32). The complete list of hypotheses for the interpretation of utterance 3-3=6 is given in (33).

```
(31) (:OBLIGATION [SYSHUM]
       (:LF-EXISTS ?v6142*T-BOXCAR [X244] NIL
         (:LF-EXISTS ?L*T-MOVE [E236]
           (:APPLY
             (:LAMBDA ?E*T-MOVE
               (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                     (:ROLE ?E*T-MOVE :R-OBJECT ?v6142*T-BOXCAR)
                     (:ROLE ?E*T-MOVE :R-DST [CORNING])))
             ?L*T-MOVE)
           (:OCCURS ?L*T-MOVE)))
       [E247])

(32) (:LF-EXISTS ?P*T-PLAN ?DM*T-ANYTHING NIL
       (:GOAL
         (:LF-EXISTS ?v6229*T-BOXCAR [X244] NIL
           (:LF-EXISTS ?L*T-MOVE [E236]
             (:APPLY
               (:LAMBDA ?E*T-MOVE
                 (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                       (:ROLE ?E*T-MOVE :R-OBJECT ?v6229*T-BOXCAR)
                       (:ROLE ?E*T-MOVE :R-DST [CORNING])))
               ?L*T-MOVE)
             (:OCCURS ?L*T-MOVE)))
         ?P*T-PLAN))

(33) (:SEQUENCE (:SURF-INTERP [CE249] [ST-ACCEPT-0006])
                (:SURF-INTERP [CE251]
                  (:AND (:OR (:EX-OR [ST-INFORM-0007] [ST-CHECK-0008] [ST-YNQ-0009])
                             [ST-SUGGEST-0010])
                        [ST-SUPP-INF-0011])))
```

In addition, the surface structure of this utterance yields two hypotheses about argumentation acts. The cue word in [CE249] leads to the :so act in (34), while the relative clause in [CE251] leads to the :background relation in (35).

```
(34) (:THE ?PA*T-SPEECHACT ?PADM*T-ANYTHING (:PREV-CE [CE249] ?PA*T-SPEECHACT)
       (:THE ?NA*T-SPEECHACT ?NADM*T-ANYTHING (:NEXT-CE [CE249] ?NA*T-SPEECHACT)
         (:SO ?PA*T-SPEECHACT ?NA*T-SPEECHACT)))

(35) (:BACKGROUND [ST-SUPP-INF-0011] [CE251])
```

The speech act evaluation proceeds as follows. First, [ST-ACCEPT-0006] is evaluated, in a similar manner to [ST-ACCEPT-0005], above. In this case, there are no acts which match the content in (29), so this hypothesis is ruled out[5]. Moving onto the hypotheses for [CE251], the system tries to prove (31) in the **Shared** context to test [ST-INFORM-0007]. The proof fails, and thus [ST-INFORM-0007] is considered applicable, making [ST-CHECK-0008] and [ST-YNQ-0009] irrelevant.

[5]This is because [ST-ACCEPT-0005] is assumed to be a part of the completed DU, rather than initiating its own DU and requiring acknowledgement and/or acceptance in its own right. If one were to adopt a theory such as the one in [Clark and Schaefer, 1989], in which acceptances always need acceptance, then [ST-ACCEPT-0006] would be applicable, as well.

Evaluating [ST-SUGGEST-0010] leads to an *incorporate* call to the domain plan reasoner. As with [ST-SUGGEST-0004], the system checks the **domain plan stack** to find a current plan for the suggestion. In this case, [PLAN-4719] is there, so this is used as a first possibility.

The :goal form in (31) leads to an :incorp-type of :goal, as with [ST-SUGGEST-0004], but the :so argumentation act, (34), leads to a hypothesis that this is a subgoal, subordinate to previously discussed content. (34) is evaluated by the pruner to identify the indexical events ?PA and ?NA. The restriction on ?NA is that it is the :NEXT-CE of [CE249]. The sequence information in (33) disambiguates this to [CE251], and thus this argumentation act potentially applies to [ST-SUGGEST-0010]. To further check the applicability, the system checks possibilities for ?PA, searching backwards from [CE249] for an event which had a speech act interpretation which added information to the plan ([PLAN-4719]). The only event which qualifies is [CE10] which had as one of its interpretations [ST-SUGGEST-0004], which added the information in (23). The plan graph node which was put in focus by this suggestion is then used as the :context argument for the *incorporate* call. The :background act in (35) leads to a :bg argument in the incorporate call – this information may prove useful in incorporating the suggestion. The actual call is given in (36).

(36)
```
(INCORPORATE
   (:LF-EXISTS ?v6229*T-BOXCAR [X244] NIL
      (:LF-EXISTS ?L*T-MOVE [E236]
         (:APPLY
            (:LAMBDA ?E*T-MOVE
               (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                     (:ROLE ?E*T-MOVE :R-OBJECT ?v6229*T-BOXCAR)
                     (:ROLE ?E*T-MOVE :R-DST [CORNING])))
            ?L*T-MOVE)
         (:OCCURS ?L*T-MOVE)))
   [PLAN-4719]
   :INCORP-TYPE :GOAL   :CONTEXT #EVENT-GOAL<[MOVE-COMMODITY-4779]>
   :BG (:LF-EXISTS ?v6247*T-ORANGES [X240] NIL
         (:AT ?v6247*T-ORANGES [CORNING] [E242])))
```

The domain plan reasoner recognizes an :incorp-type of :goal as a subgoal when there is also a :context argument. This call succeeds, with the plan propositions in (37) as the resulting additions. If this call had not succeeded, then the system would first have tried to check other possibilities for the ?PA argument of the :so act, potentially yielding different incorporate calls. If this did not work, the :so argumentation act would have been rejected and a call without a context argument would have been made. Finally, the system would have tried incorporating as the goal of a new plan.

(37)
```
(:PLAN-EVENT [MOVE-COMMODITY-4779] [PLAN-4719])
(:PLAN-EVENT [F-MOVE-CAR MOVE-COMMODITY-4779] [PLAN-4719])
(:PLAN-GENERATES [F-MOVE-CAR MOVE-COMMODITY-4779] [MOVE-COMMODITY-4779] [PLAN-4719])
(:PLAN-FACT
  (:AT [F-OBJECT [F-MOVE-CAR MOVE-COMMODITY-4779]]
       [F-SRC [F-MOVE-CAR MOVE-COMMODITY-4779]]
       [F-PRE1 [F-MOVE-CAR MOVE-COMMODITY-4779]])
```

```
  [PLAN-4719])
(:PLAN-ENABLES
  (:AT [F-OBJECT [F-MOVE-CAR MOVE-COMMODITY-4779]]
       [F-SRC [F-MOVE-CAR MOVE-COMMODITY-4779]]
       [F-PRE1 [F-MOVE-CAR MOVE-COMMODITY-4779]])
  [F-MOVE-CAR MOVE-COMMODITY-4779] [PLAN-4719])
(:PLAN-EVENT [MOVE-CAR-7867] [PLAN-4719])
(:PLAN-ACHIEVES [MOVE-CAR-7867]
       (:AT [F-OBJECT [F-MOVE-CAR MOVE-COMMODITY-4779]]
            [F-SRC [F-MOVE-CAR MOVE-COMMODITY-4779]]
            [F-PRE1 [F-MOVE-CAR MOVE-COMMODITY-4779]])
       [PLAN-4719])
(:AT   [F-OBJECT [F-MOVE-CAR MOVE-COMMODITY-4779]]
       [F-SRC [F-MOVE-CAR MOVE-COMMODITY-4779]]
       [F-PRE1 [F-MOVE-CAR MOVE-COMMODITY-4779]])
```

The propositions in (37) show that to incorporate (32) into the plan, the plan reasoner needed to add a move-car event which (partially) generates the move-commodity event which is the goal. This move-car event has an enabling condition that the car which is moved is at the source of the move-car before the event takes place, and this fact is in turn achieved by the move-car to Corning which is mentioned in the suggestion.

The final set of core speech acts determined to have occurred is shown in (38)a, while the final argumentation acts are shown in (38)b.

(38) a. `[ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]`

b. `(:SO [CE10] [CE251]) (:BACKGROUND [ST-SUPP-INF-0011] [CE251])`

This utterance is recognized as the initiation of a new DU, DU-2, with the User as the initiator and the the CSA-LIST from (38)a. Its belief context, `#<context SB-G8281>` is a child of **Hprop**, and is augmented with assertions of (37) from `[ST-SUGGEST-0010]`, (31) from `[ST-INFORM-0007]`, and (30) from `[ST-SUPP-INF-0011]`.

The turn-taking acts produced are shown in (39). Note that the second one is a `:keep-turn`, since the User signaled this in the input.

(39) `(KEEP-TURN MANAGER [NOW4]) (KEEP-TURN MANAGER [NOW4])`

The final set of conversation acts produced are shown in (40).

```
(40) :CSAS [ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
     :ARGS (:SO [CE10] [CE251]) (:BACKGROUND [ST-SUPP-INF-0011] [CE251])
     :GAS (:INIT MANAGER DU-2 ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]) NIL)
     :TTAS (KEEP-TURN MANAGER [NOW4]) (KEEP-TURN MANAGER [NOW4])
```

The resulting discourse context after the interpretation of this utterance is shown in Figure 6.13.

If the system had the turn, then a similar process of reasoning to that in Section 6.6.3 would occur. Since the user has the turn, and there are no discourse obligations, the system simply waits for the next user utterance.

```
Turn Holder: Manager
DU Stack:
 DU-2  :INITIATOR MANAGER :STATE 1       :CONTEXT #<context SB-G8281>
       :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011])
       :PROCESSED NIL
 DU-1  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
```

Figure 6.13: Discourse Context after Utterance 3-3=6

6.6.6 Interpreting utterance 3-7

USER: there are oranges at Corning. <kt>

This utterance is labeled [CE532], with core speech act hypotheses shown in (41).

(41)
```
(:SURF-INTERP [CE532]
        (:OR (:EX-OR [ST-INFORM-0012] [ST-CHECK-0013] [ST-YNQ-0014])
             [ST-SUGGEST-0015]))
```

The contents of the informational acts is shown in (42), while the content of the suggestion is shown in (43).

(42)
```
(:LF-EXISTS ?v8920*T-ORANGES [X527] NIL
            (:AT ?v8920*T-ORANGES [CORNING] [E529]))
```

(43)
```
(:THE ?P*T-PLAN ?DM*T-ANYTHING (:CURRENT-PLAN ?P*T-PLAN)
        (:LF-EXISTS ?VAR*T-ORANGES [X527] NIL
            (:USES ?VAR*T-ORANGES (:AT ?VAR*T-ORANGES [CORNING] [E529])
                   ?P*T-PLAN)))
```

Evaluating [ST-INFORM-0012] in the **Shared** context involves trying to prove (42) in this context. The proof is successful, since [ORANGES-1] are at corning [NOW], and [E529] is a sub-time of [NOW]. This rules out the inform possibility. [ST-YNQ-0014] is also ruled out, since the fact can be proved in the Speaker proposed context (**Hprop**), as well. This leaves only the [ST-CHECK-0013] possibility from the informational acts.

Evaluating [ST-SUGGEST-0015] involves an incorporate call, made in context #<context SB-G8281>, the belief context associated with the top DU, and which contains the prior suggestions and results in the plan. The definite reference form in (43) is evaluated and the restriction (:CURRENT-PLAN ?P*T-PLAN) binds ?P to the top plan in the domain plan stack, [PLAN-4719]. This is a suggestion to use the oranges which are at Corning in the plan. First the dialogue manager finds out that [ORANGES-1] are the only oranges at Corning, and so asserts that, [X527], the discourse marker for the oranges referred to in this utterance, is equal to [ORANGES-1]. Then, the call in (44) is made to the domain plan reasoner. This :incorp-type is of type :role-filler, meaning that this group of oranges should fill some role in an event in the plan.

```
(44) (INCORPORATE (:USE-OBJECT [X527])
                  [PLAN-4719]
                  :INCORP-TYPE :ROLE-FILLER :FOCUS [X527]
                  :BG (:AT [X527] [CORNING] [E529]))
```

The call is successful, returning the propositions in (45), which states that these oranges are the commodity role of the goal MOVE-COMMODITY event.

```
(45) (:PLAN-EVENT [MOVE-COMMODITY-4779] [PLAN-4719])
     (:PLAN-FACT (:AND (:EQ [X527] [F-COMMODITY MOVE-COMMODITY-4779])) [PLAN-4719])
     (:PLAN-SUPPORTS (:AND (:EQ [X527] [F-COMMODITY MOVE-COMMODITY-4779]))
                     [MOVE-COMMODITY-4779] [PLAN-4719])
     (:PLAN-PREMISE (:AND (:EQ [X527] [F-COMMODITY MOVE-COMMODITY-4779])) [PLAN-4719])
     (:AND (:EQ [X527] [F-COMMODITY MOVE-COMMODITY-4779]))
```

The validated core speech acts for this utterance are: [ST-CHECK-0013] and [ST-SUGGEST-0015].

This utterance is recognized as a :cont of the top DU, DU-2, adding the core speech acts to the CSA-list for this DU. The turn-taking acts are the same as in (39). The final conversation acts are shown in (46). The resulting discourse context is shown in Figure 6.14.

```
(46) :CSAS [ST-CHECK-0013] [ST-SUGGEST-0015] :ARGS NIL
     :GAS (:CONT MANAGER DU-2 ([ST-CHECK-0013] [ST-SUGGEST-0015]) NIL)
     :TTAS (KEEP-TURN MANAGER [NOW5]) (KEEP-TURN MANAGER [NOW5])
```

Turn Holder: Manager
DU Stack:

```
DU-2  :INITIATOR MANAGER :STATE 1       :CONTEXT #<context SB-G8281>
      :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                 [ST-CHECK-0013] [ST-SUGGEST-0015])   :PROCESSED NIL
DU-1  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G4940>
      :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
```

Figure 6.14: Discourse Context after Utterance 3-7

The system again waits for the User's next utterance.

6.6.7 Interpreting utterance 3-8

USER: right?

This utterance is labeled [CE608], and the only core speech act hypothesis is [ST-CHECK-0016], with content given in (47). This means that the utterance is checking whether the content of some previous informational speech act is correct.

```
(47) (:THE ?CE*T-INF-SA ?CEDM*T-ANYTHING (:PREV-ACT [CE608] ?CE*T-INF-SA)
           (:RIGHT-CORRECT ?CE*T-INF-SA))
```

This act is evaluated and accepted with ?CE set to [ST-CHECK-0013], the surviving informational hypothesis from the previous conversational event. The grounding act recognized for this utterance is a :repair. The repair is of the top DU, DU-2 and is using this new [ST-CHECK-0016] to replace the informational act it is checking, [ST-CHECK-0013]. This has no real effect on the conversation, since the previous interpretation was already a check, but if the prior interpretation had picked out the inform or ynq possibilities, they would have been changed to a check. This utterance also releases the turn back to the system. The conversation acts for this utterance are shown in (48), and the resulting discourse state is shown in Figure 6.15.

(48)
```
:CSAS [ST-CHECK-0016] :ARGS NIL
:GAS (:REPAIR MANAGER DU-2 ([ST-CHECK-0016]) ([ST-CHECK-0013]))
:TTAS (KEEP-TURN MANAGER [NOW6]) (RELEASE-TURN MANAGER [NOW6])
```

Turn Holder: System
DU Stack:

```
DU-2  :INITIATOR MANAGER :STATE 1        :CONTEXT #<context SB-G8281>
      :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                 [ST-SUGGEST-0015] [ST-CHECK-0016])    :PROCESSED NIL
DU-1  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G4940>
      :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
```

Figure 6.15: Discourse Context after Utterance 3-8

6.6.8 Acting after utterance 3-8

The discourse actor now decides to acknowledge the previous utterances and ground DU-2. In addition to adding this acknowledgement to the **intended speech acts**, the core speech acts from this DU will be processed and their effects added to the main discourse context. Most of the acts will just add their effects to the speaker proposed space, **Hprop**, but [ST-CHECK-0016] will add a discourse obligation to respond to the check. The resulting discourse context is shown in Figure 6.16.

Now the system will address its obligation, which adds the intended speech act in (49):

(49)
```
(:INFORM-IF (:FOCUS :RIGHT (:RIGHT-CORRECT [ST-CHECK-0013]))
            (:OBLIGATION (:CHECK-IF (:RIGHT-CORRECT [ST-CHECK-0013]))))
```

The system now generates utterance 4:

SYSTEM: Right.

6.6.9 Interpreting utterance 4

This utterance is also fed through the language interpretation modules, and the core speech act hypotheses are given in (50).

```
Discourse Obligations: (:CHECK-IF (:RIGHT-CORRECT [ST-CHECK-0013]))
Turn Holder: System
Intended Speech Acts:

 (:ACK ([ST-CHECK-0016] [ST-SUGGEST-0015] [ST-SUPP-INF-0011] [ST-SUGGEST-0010]
        [ST-INFORM-0007]) GROUNDING)
DU Stack:
 DU-2  :INITIATOR MANAGER :STATE 1        :CONTEXT #<context SB-G8281>
       :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016])
       :PROCESSED ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016])
 DU-1  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
Unaccepted Proposals: [ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                      [ST-SUGGEST-0015]
```

Figure 6.16: Discourse Context after deciding to acknowledge

(50) `(:SURF-INTERP [CE661] (:OR [ST-INFORM-0018] [ST-ACCEPT-0017]))`

The content of [ST-ACCEPT-0017] is the same as the content of [ST-ACCEPT-0005] given in (26). The content of [ST-INFORM-0018] is given in (51).

(51) `(:THE ?CE*T-INF-SA ?CEDM*T-ANYTHING (:PREV-ACT [CE661] ?CE*T-INF-SA)`
`(:RIGHT-CORRECT ?CE*T-INF-SA))`

Evaluating [ST-INFORM-0018] involves querying whether there is some informational act which is :RIGHT-CORRECT. It turns out that there is an informational act in the previous utterance, namely [ST-CHECK-0016]. This is, indeed :RIGHT-CORRECT, because [ST-CHECK-0013] is – there are oranges at Corning. Thus the inform reading is ruled out.

Evaluating [ST-ACCEPT-0017] turns up a list of [ST-SUGGEST-0015], [ST-SUPP-INF-0011], [ST-SUGGEST-0010], and [ST-INFORM-0007], which are all accepted by this act. Thus [ST-ACCEPT-0017] is the only validated core speech act.

This utterance is also seen as an :ack of the top DU, DU-2, adding [ST-ACCEPT-0017] to this DU. The turn-taking acts are the same as for utterance 2, shown in (27). The conversation acts for this utterance are shown in (52)

(52)
```
:CSAS [ST-ACCEPT-0017] :ARGS NIL
:GAS (:ACK SYSTEM DU-2 ([ST-ACCEPT-0017]) NIL)
:TTAS (KEEP-TURN SYSTEM [NOW7]) (RELEASE-TURN SYSTEM [NOW7])
```

Updating based on the acknowledge will cause the discourse context to be updated with the effects of the unprocessed acts in DU-2, namely [ST-ACCEPT-0017]. This, in turn, will cause various propositions to be moved to the **Shared** context, namely those from [ST-INFORM-0007] shown in (31), those from [ST-SUGGEST-0010] shown in (37), and those from [ST-SUGGEST-0015] shown in (45).

```
Turn Holder: Manager
DU Stack:
 DU-2  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G8281>
       :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
       :PROCESSED ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
 DU-1  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
```

Figure 6.17: Discourse Context after utterance 4

The resulting discourse context is shown in Figure 6.17.

Since there are no longer outstanding obligations, the system waits for the user's next utterance.

6.6.10 The rest of the dialogue

From this point, we give only a very cursory overview of some of the features of the processing. The completed trace is given in Appendix A.

Utterances in turns 5 and 7 have similar interpretations to those in turn 3, and similar deliberations lead to the system responses 6 and 8. Note that after the `:init` in utterance 7-1=2, `DU-1` drops off the bottom of the DU stack. The core speech acts from this utterance can no longer be influenced by successive grounding acts. Any change in this material must be performed by reintroducing the material in new DUs.

The reasoning leading up to utterance 14 is similar to that leading to utterance 2. Here the user is suggesting domain actions to help lead to the goal, and the system, when it gets the turn, acknowledges and accepts this suggestion. Utterances 15-2=4, 15-5=7, and 15-8=10 are interpreted as requests because of the imperative surface structure. The discourse obligation to address the request is incurred only when the system decides to acknowledge the utterances and ground them. After the decision to acknowledge, the obligations are incurred, and the system then addresses the requests, deciding to accept them all, and adding intentions to perform an accept speech act, which is then produced as 16.

Utterance 17 is interpreted as a request for evaluation of the plan. This is a *conversational level* rather than domain-level request, and thus does not require a call to the domain-plan reasoner to *incorporate* the content. When the system decided to acknowledge, this creates a discourse obligation to address the request. The system considers this (invoking the domain plan reasoner to search the plan for problems or incomplete parts) and decides that the plan will work, and so decides to perform the requested action – an evaluation speech act. This is then generated as 18-3. 18-3 is both an `:init` of `DU-8`, as well as an `:ack` of `DU-7`.

6.7 Discussion

There are several differences between the theoretical analysis of conversation acts given in Section 4.3 and the behavior of the TRAINS system given in the previous section and Appendix A. Some of these are trivially due to to the simplifications made to the dialogue engaged in by the implementation, while others result from differences in the detail of the analysis or differing choices made about ambiguous aspects. We will mention a few of these here. First, in the implementation, the "right?" utterances coming from 3.8, 5.2, and 7.3 are treated as `repair` acts rather than as `:reqAck` acts. As mentioned in Section 4.3, the analysis could easily go either way. The only way to really rule out the `:repair` option is to assume that there is nothing to repair, yet we frequently see repairs in which something that was conveyed implicitly is made explicit. Given the declarative mood of the previous utterances, the determination that they were `check` acts was due to contextual reasoning, so it is not easy to rule out the possibility that the "right?" tags are making the act-type more explicit.

Another difference is in the number of total DUs present. Both the analysis Section 4.3 and the implementation assume, contrary to Clark and Schaefer, that acknowledgements, qua acknowledgements, do not themselves need acknowledging, and are not in and of themselves *presentations*, even though they might occur as part of the same utterance which *does* start a new presentation. This principle is also taken one step further so that simple one-word utterances which have as their only act function to `accept` or `reject` other speech acts in a DU which they are also acknowledging, are not considered to `initiate` a new DU, and the `accept` or `reject` acts are seen as part of the old DU. In the implementation, the supposed `inform-if` interpretation provided for 4, 6, and 8 were ruled out because the content was already known, leaving only an `accept` act at the core speech act level; thus no `:init` was assumed.[6]

In addition, the core speech act analysis in the implementation is slightly more precise, adding in supplementary inform and suggestion acts for subordinate clauses, and treating the imperatives as requests rather than suggestions. Finally, the set of argumentation acts discussed are somewhat different. This level is the least well understood, and argumentation acts were made explicit only when they directly aided other aspects of the analysis by bringing to bear surface information.

The inclusion into a language conversation system of a component that tracks grounding has both benefits and new challenges that must be overcome. For one thing, it significantly increases the complexity of the context that must be maintained – responses in conversation do not generally proceed in a completely serial manner in which, first, previous material is grounded, followed by attention to introducing new material. For the vast majority of utterances, some function other than grounding is also addressed. In order to deal with these other functions, a system must be able to perform hypothetical reasoning about what the discourse context will be *if* proposed grounding

[6] Actually these utterances should probably be performing something like a `confirm` act which would stand in the relation to an `inform` that a `check` holds to a `ynq`. It would not be ruled out if the information were already known, and would be the expected reply to a `check`. Such an act could also probably be just added to the old DU, as are `accepts`.

acts are performed. If the grounding acts are not actually performed, then the old state will have to be used for future updates. This phenomenon stretches to the limit the strategy employed in the TRAINS dialogue manager of *reacting* to action and the state of the context, and planning only when necessary. The solution taken here is to have an *internal* counterpart to an acknowledgement which is performed when the system thinks it has understood the user's utterances and intends to acknowledge. At this point the context is updated, and deliberation continues until there is more to say. This strategy could run into trouble if the system is not able to acknowledge after all. In this case, the effects of the acknowledged actions would have to be backed up to the previous state.

7 Towards a Formal Theory of Plan Execution

Grounding acts are all involved with constructing local mutual understanding in the form of discourse units. As many of their names indicate (e.g., initiate, continue, repair, cancel), much of their semantics will be based on the general semantics of the construction of action. Thus in order to provide a semantics for the grounding acts, it will be helpful to illustrate the basics of a semantics of plan and action execution in general and show how it applies to the specific case of grounding.

This chapter describes a formal theory of plan execution which allows for the characterization of the performance of sub-actions as to how they relate to a larger action or plan in support of which they were performed. These characterizations include beginning, continuing, completing, cancelling, and repairing. The grounding acts can thus be seen as an application of these general relations to the domain of grounding.

Section 7.1 gives an overview of some of the requirements of a theory of plan execution which can model grounding acts. Section 7.2 informally describes the types of attitudes and processes that a plan-executing agent will undergo. Section 7.3 presents the basic single-agent theory of plan execution. This theory is then extended in the next two sections to include the execution of sub-plans (Section 7.4) and the execution of multi-agent plans (Section 7.5). Section 7.6 presents definitions of the grounding acts within this framework, making reference to a general multi-agent communication plan.

7.1 Requirements for a Theory of Plan Execution

Pollack and others have mentioned the need for distinguishing between plans as "recipes" for actions and the mental state of *having a plan*. She presents a definition of having a *simple plan* [Pollack, 1986; Pollack, 1990] in terms of beliefs and intentions that an agent has about the recipe's constituent actions. This formulation was modified by Grosz and Sidner to represent a *Shared Plan* held by mutliple agents [Grosz and Sidner, 1990]. While these formulations represent a significant increase in sophistication over previous frameworks for intended plan recognition, they are still inadequate for modelling a common plan inference situation – one in which the agent has begun executing the plan. One of the requirements for having a simple or shared plan is that the agent(s) intend all of the actions in a plan. But consider a case in which the first action has

already been performed. Clearly the agent does not intend this action any more, yet it is still a part of the plan.

One might try to amend such a theory by claiming that the intended plan consists of just those parts of the original recipe which have not yet been performed (and about which agents still have intentions). This formulation would have a number of deficiencies, however. First, it would not allow for a convenient analysis of repair, since agents often repair actions which have already been performed. Also, it will not allow representation of the roles an action plays towards plan execution, since it would be impossible to distinguish the continuation of an ongoing plan from the initiation of a new plan.

It is often useful to be able to characterize, in an abstract sense, the role a particular performance plays in the execution of a plan. In cases of intended recognition (in which the performing agent intends for the observing agent to recognize her plan) such as natural language communication, the performing agent often gives signals as to how the current action relates to plans which the observing agent should infer: whether the current performance begins a new plan, or continues, completes, repairs or cancels a pre-existing plan. For example, uttering the English clue word "oops" indicates a belief that a previous action failed. Similarly, researchers have found prosodic markings for repairs [Levelt and Cutler, 1983] and non-finality [McLemore, 1991].

7.2 A Sketch of Rational, Plan-based Behavior

Before proceeding with a formal account of plan execution, it will be helpful to informally present a sketch of the mental attitudes that a plan-executing agent will have and the deliberative processes that it will undergo. We choose a model similar to the BDI model of Bratman *et al.* [1988].

We start with **beliefs** and **desires**. From the desires (and with reference to beliefs and other goals and intentions), the agent will *deliberate* and choose a set of **goals**: conditions that the agent will try to achieve. *Planning* or *means-ends reasoning* will lead to the formation or selection of a **plan recipe** designed to meet the goals. We view recipes as consisting of a set of actions coupled with a set of constraints relating various properties of these actions (e.g. constraints on relative timing or objects and locations of actions, preconditions or effects represented as events or states which must hold over times related to the times of actions). The agent then can *adopt* or *commit to* plans. **Plans** are treated as individuals, the structure of which can be modified through time. At any given moment in time, a particular plan will correspond to a particular plan recipe. This plan adoption will lead to **commitment** to achieve (or maintain) the constraints of the plan and **intention** to perform the actions in the plan.

When an agent has committed to a plan, she can *try* to perform an action in that plan. She can also *observe* the situation she finds herself in, perhaps revising her beliefs and desires. She can also *replan*, revising the plans she is executing to correspond to different plan recipes, and thus changing her commitments and intentions. We distinguish two types of plan revision: *plan elaboration*, in which additional actions (e.g., decompo-

sitions of non-primitive actions) or constraints are added to a plan, and *plan repair*, in which some of the actions or constraints are removed. We can also distinguish the action of *plan repair* from *changing plans*. In both cases, the intentions and commitments of an agent change, but in the former case, the agent is changing the contents of a plan which she continues to execute, whereas in the latter, the agent drops all intentions and adopts a new set.

From the point at which a plan is adopted to the point at which the intentions and commitments are dropped, we say that the agent is *executing* the plan.

We can distinguish at least three distinct notions of the culmination of plan:

Action Completion – all actions in the plan have been performed.

Successful Completion – all actions in the plan have been performed and all of the constraints have been met as well.

Goal Satisfaction – the goals which motivated adoption of the plan have been achieved.

Note that Successful Completion and Goal Satisfaction are somewhat independent. It may be the case that the goals are met before the plan is completed (e.g., imagine a plan to open an elevator door by pushing a button: the goal is to get the elevator door open, but suppose it opens by itself when someone else gets out). Depending on the plan adoption procedures, a plan might also be successfully completed without having satisfied the goals. A plan is an entity independent from the goals for which it was adopted, and the same plan could be used to try to achieve many different sets of goals. If a goal is not a constraint of the plan, it may be the case that the plan is successfully executed but the goals are not met. In a distributed control scenario, a plan executor may not have access to actual goals and may simply adopt particular plans under orders of a superior.

A plan executor will monitor the success of the plans it is executing, and engage in a repair if the plan is not successful. But the plan executor will also need to monitor goals: plan repair might also be warranted if the situation changes. This could include eliminating unnecessary actions if the goals are met though events external to the plan itself.

7.3 Single-Agent Plan Execution

For simplicity, we will begin the formalization with a single-agent theory of plan execution. Although the grounding process, like much of language behavior is multi-agent action, many of the complexities will be inherent in a single-agent theory. In Section 7.5 we will show how to approach a multi-agent theory through combination of single-agent plan executions.

7.3.1 Basic Ontology

We start with a Situation Theory, similar to that of Devlin [1991]. Situations are individuals, and also provide bases for the truth or falsity of propositions. We will notate general situations by the letter S, perhaps subscripted. For situation S and proposition ϕ the notation "S $\models\phi$" means that the proposition ϕ is supported by situation S. Situations are also composable into other situations, forming a lattice structure. Following Devlin, we overload the operator "$\subseteq$" to include the "part of" relation between two situations as well as the usual subset relation between two sets. $S_1 \subseteq S_2$ means that situation S_1 is a part of situation S_2.

An interesting class of situations will be those corresponding to "happenings", which we will term *occurrences* and notate with lowercase Greek letters. A subclass of these, occurrences which are *caused* by the intentional activity of agents, will be termed *executions*. The key point of an execution is not that it has particular properties intended by a performing agent, but that intentional activity was instrumental in the performance.

Also important are abstractions over executions. We will term these *actions*, and it is these that are components of mentalistic attitudes such as intentions and plans. While a particular execution has many features, only a small subset of them are actually intended. What we call *actions* are sometimes called *action types* [Goldman, 1970] or *activities* [Balkanski, 1990].

We use the symbol "$\triangleright$" to represent the "realizes" or "is characterized by" relation between an execution and an action. We define an *Occurs* predicate over actions which is true, iff there is some execution which realizes that action:

Definition 1 $\mathrm{S} \models \mathrm{Occurs(a)} \equiv \exists\alpha : \alpha \subseteq \mathrm{S} \wedge \mathrm{S} \models \alpha \triangleright \mathrm{a}$

An execution could realize unrelated action types, and whenever an execution realizes an action type, it always realizes a more abstract action type as well. Section 7.4.1 discuses action abstraction in more detail. As an example, suppose there is some execution α which realizes a MakeSpaghetti action: $\alpha \triangleright \mathrm{MakeSpaghetti(Agt, time)}$. It would then also be the case that this same execution realizes a MakeNoodles action: $\alpha \triangleright \mathrm{MakeNoodles(Agt, time)}$. It might also be the case that this same execution also realizes some unrelated action, say $\alpha \triangleright \mathrm{WakeUpDog(Agt, time)}$. In this case, we have two separate actions performed in the same execution.

Plans, Recipes, and Mental Attitudes

We treat plans as individuals; they are abstract objects whose attributes can be modified through time, just as physical objects can. We use the term *recipe* to denote the functional characteristics of a plan at a particular time. A recipe is a set of actions coupled with a set of constraints relating various properties of these actions (e.g., constraints on relative timing, objects and locations of actions, goals and preconditions represented as events or states which must hold over times related to the times of actions). We denote

the set of actions of a recipe, R, by **Actions**(R). The set of constraints on a recipe will be denoted by **Constraints**(R)[1].

The agent of an action, a_i, in a recipe, R, is denoted by **Agt**(a_i, R). For single-agent recipes, all actions in the recipe will have the same agent parameter.

The parameterized time of an action, a_i, in a recipe, R, is denoted by **Time**(a_i, R). This will in general be a constrained variable for most recipes.

We can define a partial temporal order among actions in a recipe, according to their temporal restrictions.

Definition 2 $a_i \overset{R}{\prec} a_j$ *iff* **Before**(**Time**(a_i, R),**Time**(a_j, R)).[2]

A recipe can be said to be performed in a situation if all the actions have been performed and all of the constraints have been met. Formally,

Definition 3 **Performedin**(R, S) *iff*

$$\forall a_i : a_i \in \textbf{Actions}(R) \supset \exists \alpha : \alpha \subseteq S \wedge S \models \alpha \rhd a_i \ \wedge$$
$$\forall C : C \in \textbf{Constraints}(R) \supset S \models C$$

A plan may be reflected by different recipes at different times, changing as an agent modifies the plan. We denote the recipe of a plan, P, at time t as **Recipe**(P, t).

There are several other mentalistic notions which we will assume, but not attempt to define or axiomatize. These are: belief, intention, commitment, and intentional-action. *Belief* is a relation between an agent and a proposition, and we will use the notation[3] **Bel**(Agt, ϕ), to represent that agent **Agt** believes ϕ. We represent (future directed) *intention* as having two parameters other than the agent: an action which the agent intends to perform, and a plan which this action is intended to (in part) achieve – **Intends**(Agt, a, P). This is roughly equivalent to the expression Int(Agt,By(a,P)) used by Pollack and Grosz and Sidner. *Commitment* is a weaker notion than intention. If an agent is committed to a proposition, this commitment will influence future deliberation to avoid actions which will result in the negation of the proposition holding, and if the agent comes to believe that the proposition will not hold, this will motivate deliberation and potential adoption of new intentions. Unlike intentions, however, commitments will not, in and of themselves, involve future action by the agent (although they may serve as the source for the adoption of intentions to perform actions, as well as being the metric by which the success or failure of such actions is judged). We represent commitment as a relation between an agent and a proposition, **Committed**(Agt, ϕ). The **committed**

[1]Many of these constraints will be implicit in any *Representation* of a recipe, e.g. as shared variables in two separate actions, or domain constraints on possible recipes.

[2]Using the temporal relations of [Allen, 1983a], this would be **Time**(a_i, R) (m <) **Time**(a_j, R). For simplicity we are currently ignoring the possibility of temporally overlapping sub-actions. We won't be hampered by this assumption if simultaneous executions can be combined and considered atomic.

[3]For simplicity, we generally omit the situational and temporal parameters of belief, commitment and intention. We will sometimes use an extra temporal parameter, representing the time of the attitude, where that is relevant such as in Axiom 2, below.

relation is similar to the *Int.Th* operator of Grosz and Kraus [1993], while **intends** is like their *Int.To*. Finally, we represent present directed intention as an abstract action in its own right, called *try*. $\alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$ means that the occurrence α is characterized by **Agt** attempting to do action **a** (intentionally) as part of executing Plan **P**.

We can now formally define an execution as an occurrence for which there is some agent and plan such that the occurrence realizes the agent trying to perform an action in the plan: α is an execution iff $\exists \mathrm{Agt}, \mathrm{a}, \mathrm{P} : \alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$.

Plan Execution Situations

Now we are in a position to introduce the central theoretical construct of this chapter, the Plan Execution Situation, roughly a generalization of Pollack's notion of an agent *having* a plan [Pollack, 1986; Pollack, 1990], translated to the logic of situations.

A *Plan Execution Situation* (PES) is a situation in which a plan is being executed. It is a piece of the world containing a plan and the relevant agents, plan, objects, and events that make up executions or attempted executions of the plan. Each PES will have a designated current reference time, **now**. We represent a plan execution situation (named PE) as a 5-tuple, $[\mathrm{Agt}, \mathrm{P_{PE}}, \mathrm{E_{PE}}, \mathrm{Bind_{PE}}, \mathrm{S_{PE}}]$ where:

Agt represents the agent that is executing the plan.

$\mathrm{P_{PE}}$ represents the plan that this is the execution *of*.

$\mathrm{E_{PE}}$ represents the set of executions which have been performed in executing the plan. I.e., $\alpha \in \mathrm{E_{PE}}$ iff $\exists \mathrm{a} : \alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P_{PE}})$.

$\mathrm{Bind_{PE}}$ represents the *instantiation* (partial) function from actions of the plan to performed executions. For each action in the plan, its instantiation is an execution which the agent believes realizes that action. We will generally view this function as a set of relationships of the form: $\mathrm{a_i} \rightsquigarrow \alpha_\mathrm{i}$ where $\mathrm{a_i} \in \mathbf{Actions}(\mathbf{Recipe}(\mathrm{P_{PE}}, \mathbf{now_{PE}}))$, $\alpha_\mathrm{i} \in \mathrm{E_{PE}}$.[4] We will use a superscript version, $\mathrm{a_i} \overset{\mathrm{PE}}{\rightsquigarrow} \alpha_\mathrm{i}$ as shorthand to indicate that $\mathrm{a_i} \rightsquigarrow \alpha_\mathrm{i} \in \mathrm{Bind_{PE}}$.

$\mathrm{S_{PE}}$ represents the execution status, either 1 or 0, representing whether or not the plan is being executed.

Since each PES has its designated **now** time we will generally omit the temporal parameters from the recipe designators for the PES's plan. Thus $\mathbf{Recipe}(\mathrm{P_{PE}})$ will be used as shorthand for $\mathbf{Recipe}(\mathrm{P_{PE}}, \mathbf{now_{PE}})$. Similarly, we will use $\mathbf{Actions}(\mathrm{P_{PE}})$ to refer to the set of actions in the plan's recipe at the **now** time: $\mathbf{Actions}(\mathbf{Recipe}(\mathrm{P_{PE}}, \mathbf{now_{PE}}))$, and $\mathbf{Constraints}(\mathrm{P_{PE}})$ will stand for the set of constraints of the recipe at the **now** time: $\mathbf{Constraints}(\mathbf{Recipe}(\mathrm{P_{PE}}, \mathbf{now_{PE}}))$.

Two predicates will be useful for classifying actions in a PES's plan as to whether they have been performed or not:

[4] Actually, it might take several executions to instantiate an abstract action, depending on how executions are divided up, so the range should actually be a subset of $\mathrm{E_{PE}}$ rather than a single element.

Definition 4 **Instantiated**(a, PE) *iff* $a \in \textbf{Actions}(P_{PE}) \wedge \exists_{\alpha \in E_{PE}} a \overset{PE}{\rightsquigarrow} \alpha$

Definition 5 **Uninstantiated**(a, PE) *iff* $a \in \textbf{Actions}(P_{PE}) \wedge \neg\textbf{Instantiated}(a, PE)$

Note that an action that is not in the plan at all is neither instantiated or uninstantiated with respect to the PES.

An agent is executing a plan if (1) the agent intends to perform all uninstantiated actions in support of the plan, (2) for each instantiated action, the agent believes that the execution which the action is instantiated to realizes that action, and (3) the agent is committed to each of the constraints in a plan, as well as the propositions that are the objects of the beliefs in (2).

With regards to particular situations, an agent, Agt, is executing a plan P in a situation S if and only if there is a PES that is part of situation S that has P as its plan and execution status 1. If there is a PES with P as its plan and execution status 0, then S is a situation which includes Agt not executing P. Some situations will not have as a part either the execution or non-execution of a plan.

A PES is part of a situation if all the executions of the PES are part of the situation, the situation provides evidence that the agent believes that the instantiated actions are realized by the executions to which they are bound, and the situation provides complete information about the intentions of the agent with respect to the uninstantiated actions in the plan. If the status is 1 (representing a plan that *is* being executed), then in order for the PES to be part of a situation, the situation must provide evidence that the agent intends each of the uninstantiated actions. If, on the other hand, the status is 0 (representing a plan that is *not* being executed), then that PES is part of the situation only if the situation provides evidence that the uninstantiated actions are *not* intended (as part of the plan). Formally,

Axiom 1 $\forall PE, S : PE \subseteq S \equiv$
$\forall_{\alpha \in E_{PE}} \alpha \subseteq S \wedge$
$\forall [a_i \rightsquigarrow \alpha_i \in \text{Bind}_{PE}]\ S \models \textbf{Bel}(Agt, \alpha_i \rhd a_i) \wedge$
$(S_{PE} = 0) \wedge \forall a : \textbf{Uninstantiated}(a, PE) \supset S \models \neg\textbf{Intends}(Agt, a, P_{PE}) \vee$
$[(S_{PE} = 1) \wedge \forall a : \textbf{Uninstantiated}(a, PE) \supset S \models \textbf{Intends}(Agt, a, P_{PE}) \wedge$
$\forall C : C \in \textbf{Constraints}(P_{PE}) \supset S \models \textbf{Committed}(Agt, C) \wedge$
$\forall [a_i \rightsquigarrow \alpha_i \in \text{Bind}_{PE}]\ S \models \textbf{Committed}(Agt, \alpha_i \rhd a_i)]$

Some situations will not support either an intention or the negation of the intention (i.e., $\neg[S \models \textbf{Intends}(Agt, a, P_{PE})] \wedge \neg[S \models \neg\textbf{Intends}(Agt, a, P_{PE})]$). These situations will not contain any plan execution situation, PE, which has **a** as one of the uninstantiated actions of P_{PE}.

There may be a number of plan execution situations which are part of a given situation. There will be at least one for each different plan being executed, but there will also be PES's for sub-plans of any plan being executed (see section 7.4).

We define the remaining acts of a plan execution situation, **RActs**(PE) as those actions in the plan which have not been instantiated by any of the executions – these

are the actions which still remain to be executed in order for the plan execution situation to be completed. Formally, $\mathbf{RActs}(\mathrm{PE}) \stackrel{def}{=} \{\mathrm{a_i} \mid \mathbf{Uninstantiated}(\mathrm{a_i}, \mathrm{PE})\}$

A Plan Execution Situation, PE, is **action-completed** if all of the actions in the plan are instantiated by executions. Formally, $\mathbf{ACompleted}(\mathrm{PE})$ iff $\mathbf{RActs}(\mathrm{PE}) = \{\}$

A PES is **successful** in a situation if it is completed and all of its constraints have been met: $\mathbf{Successful}(\mathrm{PE}, \mathrm{S})$ iff
$\mathbf{ACompleted}(\mathrm{PE}) \wedge PE \subseteq S \wedge \forall C : C \in \mathbf{Constraints}(\mathrm{P_{PE}}) \supset S \models C$

7.3.2 Updating Plan Execution Situations

The *updating* of a situation by an occurrence is a situation that includes both the old situation as well as the occurrence. Generally this will involve adding an occurrence which happens at the **now** point of the situation, and the **now** point is then updated to immediately after the time of the occurrence. We introduce a situation updating function, $\mathbf{Update}(\alpha, \mathrm{S})$ which takes as parameters an occurrence and a situation, and returns the updated situation. A specialization of this is when the occurrence is an execution relevant to a plan execution situation.

There are several ways in which an execution α may update a plan execution situation PE. The simplest is if the execution is intended to instantiate one or more of the actions in $\mathbf{RActs}(\mathrm{PE})$. We define a predicate **DoNext** over an execution and a PES which is true when there is a second PES which is an update of the first with the execution and binding relationship added:

Definition 6 $\mathbf{DoNext}(\alpha, \mathrm{PE})$ *iff*[5]

$$\begin{array}{l} \exists \mathrm{a} \in \mathbf{RActs}(\mathrm{PE}) : \alpha \rhd \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P_{PE}}) \wedge \\ \quad \exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge \mathbf{Recipe}(\mathrm{P_{PE'}}) = \mathbf{Recipe}(\mathrm{P_{PE}}) \wedge \\ \quad\quad \mathrm{E_{PE'}} = \mathrm{E_{PE}} \cup \{\alpha\} \wedge \mathrm{Bind_{PE'}} = \mathrm{Bind_{PE}} \cup \{\mathrm{a} \rightsquigarrow \alpha\} \end{array}$$

Sub-Action Relations

We may further classify executions as to what role they play in the progression of the plan execution. The first execution of an action in the plan of a PES will *begin* the execution of the plan:

Definition 7 $\mathbf{BEGINS}(\alpha, \mathrm{PE})$ *iff* $\mathbf{DoNext}(\alpha, \mathrm{PE}) \wedge \mathrm{E_{PE}} = \{\}$

Subsequent executions will *continue* the plan execution:

Definition 8 $\mathbf{CONTINUES}(\alpha, \mathrm{PE})$ *iff* $\mathbf{DoNext}(\alpha, \mathrm{PE}) \wedge \mathrm{E_{PE}} \neq \{\}$

[5]For simplicity, the definition presented here allows an execution to realize only a single action in the plan. The actual definition replaces the bound action, a, with a subset of $\mathbf{RActs}(\mathrm{PE})$.

The final execution will *complete* the plan execution:

Definition 9 $\mathbf{COMPLETES}(\alpha, \mathrm{PE})$ *iff*
$\mathbf{DoNext}(\alpha, \mathrm{PE}) \wedge \mathbf{ACompleted}(\mathbf{Update}(\alpha, \mathrm{PE}))$

An execution might also instantiate all actions in the plan:

Definition 10 $\mathbf{PERFORMS}(\alpha, \mathrm{PE})$ *iff* $\mathbf{BEGINS}(\alpha, \mathrm{PE}) \wedge \mathbf{COMPLETES}(\alpha, \mathrm{PE})$

Failure and Repair

We can define a *failure* as an execution which does not realize the action which it was intended to.

Definition 11 $\mathbf{Fail}(\alpha)$ *iff* $\exists \mathrm{Agt}, \mathrm{a}, \mathrm{P} : \alpha \rhd \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P}) \wedge \neg\alpha \rhd \mathrm{a}$

What is more relevant for plan execution than a normative characterization of failure is perceived failure. That is, an execution which the performing agent takes to have been a failure. These are any executions of a PES which are not bound to an action in the plan.

Definition 12 $\mathbf{PerFail}(\alpha, \mathrm{PE})$ *iff* $\alpha \in \mathrm{E_{PE}} \wedge \neg\exists \mathrm{a_i} : (\mathrm{a_i} \in \mathbf{Actions}(\mathrm{P_{PE}}) \wedge \mathrm{a_i} \overset{\mathrm{PE}}{\rightsquigarrow} \alpha)$

If an agent believes that her action is a failure, the updated PES will include the execution, but will not have any new binding relationships.

There are at least two ways of continuing a plan execution other than simply updating the plan by performing a next action. One is to change the instantiation function such that either some action $\mathrm{a_i}$ in the plan which was previously instantiated by some some execution α is no longer instantiated by that execution, thus requiring a new execution to bind that action in any completed plan, or, conversely, that some $\mathrm{a_i}$ which was previously uninstantiated is newly bound by an execution α already part of the PES. Other instantiations in the plan execution would remain the same. We call this **ERepair**, standing for *execution repair*, since the same plan is maintained but the beliefs are changed about the success of previous action, and thus intentions about future actions are also changed.

Definition 13 $\mathbf{ERepair}(\alpha, \mathrm{PE})$ *iff* $\exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge$
$\mathbf{Recipe}(\mathrm{P_{PE'}}) = \mathbf{Recipe}(\mathrm{P_{PE}}) \wedge \mathrm{E_{PE'}} = \mathrm{E_{PE}} \wedge \mathrm{Bind_{PE'}} \neq \mathrm{Bind_{PE}}$

Another type of plan execution repair is to modify the plan itself, so that it is composed of a different recipe, though it still uses some of the same executions. This might also entail further **ERepair**, if it eliminates actions which have already been instantiated. We call this **PRepair**, standing for Plan Repair, since we are changing the plan that is being executed. In a **PRepair** the agent has changed intentions from performing the actions in the old recipe to performing actions in the new one.

Definition 14 $\mathbf{PRepair}(\alpha, \mathrm{PE})$ *iff*
$\exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge \mathbf{Recipe}(\mathrm{P}_{\mathrm{PE}'}) \neq \mathbf{Recipe}(\mathrm{P}_{\mathrm{PE}}) \wedge \mathrm{E}_{\mathrm{PE}'} = \mathrm{E}_{\mathrm{PE}}$

A plan execution may also terminate without success. An execution *cancels* a PES if it changes the execution status from one to zero. Formally:

Definition 15 $\mathbf{CANCELS}(\alpha, \mathrm{PE})$ *iff*
$\exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge \mathrm{S}_{\mathrm{PE}} = 1 \wedge \mathrm{S}_{\mathrm{PE}'} = 0$

Cancelling a PES amounts to the agent involved dropping the appropriate intentions to perform the plan by doing the specified actions. This doesn't necessarily indicate dropping the intention to achieve the goal behind the plan, and in fact the agents may start a new plan execution using the same recipe.

Note that the repair and cancellation executions themselves are not strictly part of the execution set of the plan they repair. However, they will definitely be part of any maximal situation in which the PES is a part. An agent might also have a meta-plan concerning operations on another plan. In this case, repair executions of the second plan may be part of the execution set of the PES for the execution of the meta-plan.

Some occurrences will be relevant for the eventual success or failure of a plan execution, but do not fall into any of the above categories. These will not be part of the PES as such, but will certainly be included in updates of the situation in which success is evaluated, and may be cause for an agent to undertake plan repair, especially if they affect the feasibility of the plan execution.

7.3.3 Abstract Sub-Actions

Now we are in a position to define the abstract sub-actions, **Begin**(P), **Continue**(P), **Complete**(P), **Do**(P), **Repair**(P), **Cancel**(P), which have to do with executing a plan in a situation and are based on the relationships described in Section 7.3.2. Thus, for example, the action of beginning a Plan P is performed in a given situation if that performance begins a Plan execution situation with Plan P as its plan.

Definition 16 $S \models \alpha \triangleright \mathbf{Begin}(\mathrm{P})$ *iff*
$\exists \mathrm{PE} : \mathrm{PE} \subseteq \mathrm{S} \wedge \mathrm{P}_{\mathrm{PE}} = \mathrm{P} \wedge \mathbf{BEGINS}(\alpha, \mathrm{PE})$

Definition 17 $S \models \alpha \triangleright \mathbf{Continue}(\mathrm{P})$ *iff*
$\exists \mathrm{PE} : \mathrm{PE} \subseteq \mathrm{S} \wedge \mathrm{P}_{\mathrm{PE}} = \mathrm{P} \wedge \mathbf{CONTINUES}(\alpha, \mathrm{PE})$

Definition 18 $S \models \alpha \triangleright \mathbf{Complete}(\mathrm{P})$ *iff*
$\exists \mathrm{PE} : \mathrm{PE} \subseteq \mathrm{S} \wedge \mathrm{P}_{\mathrm{PE}} = \mathrm{P} \wedge \mathbf{COMPLETES}(\alpha, \mathrm{PE})$

Definition 19 $S \models \alpha \triangleright \mathbf{Do}(\mathrm{P})$ *iff*
$\exists \mathrm{PE} : \mathrm{PE} \subseteq \mathrm{S} \wedge \mathrm{P}_{\mathrm{PE}} = \mathrm{P} \wedge \mathbf{PERFORMS}(\alpha, \mathrm{PE})$

Definition 20 $S \models \alpha \triangleright$**Repair**(P) *iff*
$\exists$PE : PE$\subseteq$S $\wedge$ P_{PE} = P $\wedge$ (**ERepair**(α, PE) $\vee$ **PRepair**(α, PE))

Definition 21 $S \models \alpha \triangleright$**Cancel**(P) *iff*
$\exists$PE : PE$\subseteq$S $\wedge$ P_{PE} = P $\wedge$ **CANCELS**(α, PE)

7.4 Sub-Plans and Sub-Plan Executions

Although plans are individuals, we can still talk about subpart relationships between plans. Complex plans (those with a recipe which has multiple actions and/or constraints) are often composed of sub-plans which correspond to simpler recipes. Since recipes are just a pair of sets (a set of actions coupled with a set of constraints), we can induce a lattice of recipes based on inclusion of actions and constraints. We will overload the $\subseteq$ operator to include the sub-recipe relationship:

Definition 22 R1 $\subseteq$ R2 iff
Actions(R1) $\subseteq$ **Actions**(R2) $\wedge$ **Constraints**(R1) $\subseteq$ **Constraints**(R2)

The join of two recipes will be a recipe with action set the union of the sets of actions of the two recipes and with constraint set the union of constraint sets of the two recipes[6]. The meet of two recipes will be the pair of the intersections of the two sets (with $\perp$ being the null recipe with no actions or constraints).

The sub-plan structure of an agent is somewhat controversial. It is not clear whether the fact that an agent intends a plan with a particular recipe entails that the agent intends a sub-plan for every sub-recipe of that recipe. Certainly some of the sub-recipes will not form natural classes in and of themselves, and would be unnatural as top-level intentions. It will, of course, be the case that all of the actions of the sub-recipe will be intended and all of the constraints will be committed to, but this does not seem to be sufficient motivation to require that an agent have a sub-plan for every sub-recipe.

Instead of attempting to define the sub-plan structure from first principles, we will simply allow for a plan to contain sub-plans and will axiomatize some of the properties of sub-plans, when they do exist. We use the relation **SubPlanOf**(P_1, P_2, t) to indicate that P_1 is a sub-plan of P_2 at time t. Whenever a plan has a sub-plan, the recipe of the sub-plan will be a sub-recipe of the recipe of the main plan:

Axiom 2 $\forall$P, P$'$: **SubPlanOf**(P$'$, P, t) $\supset$ [**Recipe**(P$'$, t) $\subseteq$ **Recipe**(P, t)]

This sub-plan relation will also be important for relating intentions to plans. If an action is intended as part of a plan and that action is also part of the sub-plan, it is intended as part of the sub-plan:

[6] Of course some of these recipes will have inconsistent constraint sets and thus will not be achievable in any world.

Axiom 3 $\forall S, Agt, a, t, P, P' : \mathbf{SubPlanOf}(P', P, t) \wedge a \in \mathbf{Actions}(\mathbf{Recipe}(P', t)) \supset$
$S \models \mathbf{Intends}(Agt, a, P, t) \equiv S \models \mathbf{Intends}(Agt, a, P', t)$

There is a similar relation for present-directed intention. If there is some execution which is characterized by an agent trying to do an action as part of a plan and the action is part of a sub-plan at the time of execution, then the execution also is characterized by the agent trying to to do the action as part of the sub-plan:

Axiom 4 $\forall \alpha, Agt, a, P, P' : \mathbf{SubPlanOf}(P', P, \text{time}(\alpha)) \wedge$
$a \in \mathbf{Actions}(\mathbf{Recipe}(P', \text{time}(\alpha))) \supset$
$\alpha \rhd \mathbf{Try}(Agt, a, P) \equiv \alpha \rhd \mathbf{Try}(Agt, a, P')$

A plan execution situation can be *part of* another. This is intuitively what is happening when agents are executing a sub-plan as part of executing a larger plan. One PES will be part of another when its plan is a sub-plan, and its executions and set of binding relationships are subsets of those of the other PES. Irrelevant executions and binding relationships will be excluded from the sub-PES, because, by the definition of a PES, only executions which are characterized by the agent trying to perform an action in the plan of the sub-PES will be included in the execution set of the sub-PES. Similarly, only binding relationships which bind executions of the sub-PES to actions in the plan of the sub-PES will be included. We will also want to add conditions to force all relevant executions and binding relationships to be part of the sub-plan.

Definition 23 $\mathbf{PartOf}(PE', PE)$ *iff* $\mathbf{SubPlanOf}(P_{PE'}, P_{PE}, \mathbf{now_{PE}}) \wedge E_{PE'} \subseteq E_{PE} \wedge$
$\text{Bind}_{PE'} \subseteq \text{Bind}_{PE} \wedge S_{PE'} = S_{PE} \wedge \mathbf{now_{PE}} = \mathbf{now_{PE'}} \wedge$
$\forall \alpha : \alpha \in E_{PE} \wedge \exists a : a \in \mathbf{Actions}(P_{PE'}) \wedge a \overset{PE}{\rightsquigarrow} \alpha \supset a \overset{PE'}{\rightsquigarrow} \alpha \wedge \alpha \in E_{PE'}$

From axiom 1, which states what it means for a PES to be part of a situation, we can prove that any situation which contains a PES with status 1 also contains any PES which is part of that PES:

Theorem 1 $\forall PE', PE, S : \mathbf{PartOf}(PE', PE) \wedge (S_{PE} = 1) \wedge (PE \subseteq S) \supset PE' \subseteq S$
Proof: *Since* PE′ *is part of PE, all of the actions, executions, and binding relationships in* PE′ *will also be in PE. Because PE is part of S, each of the executions will be part of S, but then, by transitivity of the "part of" relationship, all of the executions of* PE′ *will also be part of S. Similar reasoning shows that the belief condition holds for the binding relationships in* PE′*, as well as the commitment conditions. Because of Axiom 3, the intention conditions will hold for all of the uninstantiated actions in* $P_{PE'}$*. Thus all of the conditions are met for* PE′ *to be part of S.* □

Note that other relations are not universal. If a sub-PES is part of a situation, the super-PES might not also be. Also, if a PES with status 0 is part of a situation, a sub-PES might not be, because there might be no way to prove that the situation supports the fact that the agent doesn't intend to perform an action as part of the plan of the sub-PES.

We can now prove some facts about the co-occurrence of some of the abstract sub-actions from Section 7.3.2.

Theorem 2 *Some* **Continues** *are also* **Begins**
Claim 1 $\exists P, P', t : \mathbf{SubPlanOf}(\mathrm{P}', \mathrm{P}, \mathrm{t}) \wedge \forall \alpha, s : s \models \alpha \triangleright \mathbf{Continue}(\mathrm{P}) \supset s \models \alpha \triangleright \mathbf{Begin}(\mathrm{P}')$
Claim 2 $\exists PE, \mathrm{PE}' : \forall \alpha, s : PE \subseteq s \wedge \mathbf{CONTINUES}(\alpha, \mathrm{PE}) \supset \mathrm{PE}' \subseteq s \wedge \mathbf{BEGINS}(\alpha, \mathrm{PE}')$
Proof: *If* $s \models \alpha \triangleright \mathbf{Continue}(\mathrm{P})$ *then by Definitions 17 and 8,* $\exists \mathrm{PE} \mid \mathrm{P} = \mathrm{P}_{\mathrm{PE}} \wedge \mathrm{PE} \subseteq \mathrm{s} \wedge \mathrm{E}_{\mathrm{PE}} \neq \{\}$. *But if there is also a plan* P' *such that* $\mathbf{SubPlanOf}(\mathrm{P}', \mathrm{P}, \mathrm{t}) \wedge \mathbf{Actions}(\mathbf{Recipe}(\mathrm{P}', \mathrm{t})) = \mathbf{RActs}(\mathrm{PE})$, *then consider* PE′, *which is defined as follows:* $\mathrm{P}_{\mathrm{PE}'} = P', \mathrm{E}_{\mathrm{PE}'} = \{\}, \mathrm{Bind}_{\mathrm{PE}'} = \{\}, \mathrm{S}_{\mathrm{PE}'} = 1$.
This is a PES for executing the subplan of P which has not yet been executed, and will also be part of s if PE is. Now $\mathbf{Update}(\alpha, \mathrm{PE}') = [\mathrm{Agt}, \mathrm{P}_{\mathrm{PE}'}, \{\alpha\}, \{\mathrm{a_i} \rightsquigarrow \alpha \mid \mathrm{a_i} \in \mathrm{A}\}, 1]$ *where A is the same set of actions bound by* α *in PE. But then according to definitions 6 and 7,* $\mathbf{BEGINS}(\alpha, \mathrm{PE}')$ *and by definition 16,* $s \models \alpha \triangleright \mathbf{Begin}(\mathrm{P}_{\mathrm{PE}'})$ □

Though some **Begins** will also be **Continues** (e.g. those which are non-initial subplans), a **Begin** of any top-level plan, which is intended purely for its own sake and not in part to fulfill higher goals will not be a **Continue**.

7.4.1 Action and Recipe Abstraction

It is often also useful to consider the relationships between plans which do not contain subsets of the action and constraint set, yet which still, in some sense, are in a part-whole relationship. First, we can start with the relationship between actions which are more abstract or specialized than other actions. A more specialized action will be more constrained and will be realized by a reduced set of occurrences. We can thus present a semantic definition of specialization based on the occurrences which realize the actions: we say $\mathrm{a_1}$ specializes $\mathrm{a_2}$ ($\mathrm{a_1} \sqsubseteq \mathrm{a_2}$) iff any instance of $\mathrm{a_1}$ is also an instance of $\mathrm{a_2}$. Formally,

Definition 24 $\mathrm{a_1} \sqsubseteq \mathrm{a_2}$ *iff* $\forall \alpha : \alpha \triangleright \mathrm{a_1} \supset \alpha \triangleright \mathrm{a_2}$

Individual actions will thus form a join semilattice based on abstractness. The join of two actions is an action which abstracts both actions. $\top$ will be the *universal action* which is performed by any occurrence in which an action is performed.

We will also want a notion of recipe abstraction. As a first cut we will also use a semantic notion based on the notion of plan subsumption in [Devanbu and Litman, 1991]. Thus a recipe $\mathrm{R_1}$ specializes another recipe $\mathrm{R_2}$ iff every situation in which $\mathrm{R_1}$ has been successfully performed is also one in which $\mathrm{R_2}$ has also been successfully performed. Formally,

Definition 25 $\mathrm{R_1} \sqsubseteq \mathrm{R_2}$ *iff* $\forall S : \mathbf{Performedin}(\mathrm{R_1}, \mathrm{S}) \supset \mathbf{Performedin}(\mathrm{R_2}, \mathrm{S})$

7.5 Multi-Agent Plan Execution

In this section, we extend the notion of plan execution situations to allow the execution of multi-agent plans. Intuitively, a multi-agent plan is simply one in which actions are

performed by more than one agent. However, there are a number of complexities in relating how the situation of executing a multi-agent plan corresponds to the individual and joint attitudes of the participants. For instance, agents will only intend to do their own actions, not those of the other agents. Simple intention by each agent to perform *just* her own actions will not work either, as Grosz and Sidner [1990] convincingly show with an example of two children building a tower of blocks: the children do not each have plans to just place their own blocks over empty spaces in just the right places so the other child's blocks will fit – some awareness and orientation to the other agent is necessary. The challenge is thus to determine what the proper attitudes are for agents engaged in executing a multi-agent plan.

We will assume that, in addition to intending her own actions, an agent executing a multi-agent plan will be **committed** to the occurrence of the actions of the other agent, in effect, adding the occurrence of these actions as constraints on her single agent plan. Each agent will thus be executing a slightly different yet related single agent plan. In addition, each agent will be **obliged** to the other agents to perform her actions as part of the plan.

We represent obligation as a relation between an obligated agent, *Agt*, an obliged action, *a*, a group of agents to whom the agent is obliged, *Agts*, and a plan, *P*, which the obliged action is a part of: **Obliged**(Agt, a, Agts, P)

We can thus represent a multi-agent plan execution situation as a 5-tuple, replacing the Agent parameter with a set of agents: $[A_{PE}, P_{PE}, E_{PE}, Bind_{PE}, S_{PE}]$. The parameters of the multi-agent plan execution situation (MPES) can be related to a set of single-agent plan execution situations, PE_{A_i}, one for each agent A_i in the set of agents A_{PE}.

For each of these PE_{A_i}, the actions will be just those actions of the multi-agent plan for which A_i is the agent. Formally,

$$\forall_{A_i \in A_{PE}} \mathbf{Actions}(P_{PE_{A_i}}) = \{a_i \mid a_i \in \mathbf{Actions}(P_{PE}) \wedge \mathbf{Agt}(a_i, P_{PE}) = A_i\}$$

The set of constraints of each PE_{A_i} will be the constraints of the multi-agent plan along with the occurrence of each action in the multi-agent plan for which A_i is not the agent:

$$\forall_{A_i \in A_{PE}} \mathbf{Constraints}(P_{PE_{A_i}}) =$$
$$\mathbf{Constraints}(P_{PE}) \cup \{\mathrm{Occurs}(a_i) \mid a_i \in \mathbf{Actions}(P_{PE}) \wedge \mathbf{Agt}(a_i, P_{PE}) \neq A_i\}$$

Each $P_{PE_{A_i}}$ is a sub-plan of P_{PE}. Although it might look as if Axiom 2 is violated by adding the occurrence conditions to the $P_{PE_{A_i}}$, this is not really a problem, because we can treat the occurrence of actions in the plan as (perhaps implicit) constraints on the plan without loss of generality. Because of the truth conditions for *Occurs* in Definition 1, these constraints will already be met whenever the recipe is successful according to Definition 3. We can also add an axiom relating intentions to commitment to take care of the commitment conditions for these constraints:

Axiom 5 $\forall S, Agt, a, P : S \models \mathbf{Intends}(Agt, a, P) \supset S \models \mathbf{Committed}(Agt, \mathrm{Occurs}(a))$

The execution set of the MPES will be the union of the execution sets of each individual PES: $E_{PE} = \bigcup_{A_i \in A_{PE}} E_{PE_{A_i}}$

The set of binding relationships for the MPES will be the union of the sets of relationships of the individual PES's: $Bind_{PE} = \bigcup_{A_i \in A_{PE}} Bind_{PE_{A_i}}$

Finally, the execution status of the MPES will be the boolean product of the execution statuses of the individual PES's. That is, if all of the PES's have status 1, the status of the MPES will be 1, if any of the PES's have status 0 (meaning at least one agent is not executing her individual plan), then the multi-agent plan is not being executed:

$S_{PE} = \prod_{A_i \in A_{PE}} S_{PE_{A_i}}$

Note that an MPES with a single agent is equivalent to the sub-PES for that agent:

If $A_{PE} = \{A_1\}$ then
$\mathbf{Actions}(P_{PE}) = \mathbf{Actions}(P_{PE_{A_1}}) \wedge \mathbf{Constraints}(P_{PE}) = \mathbf{Constraints}(P_{PE_{A_1}}) \wedge$
$E_{PE} = E_{PE_{A_1}} \wedge Bind_{PE} = Bind_{PE_{A_1}} \wedge S_{PE} = S_{PE_{A_1}}$

For an MPES to be part of a situation, we need all of the constituent PES's to be part of that situation (and thus the belief, commitment and intention conditions for those PES's must hold). In addition, an agent will be obliged to all other agents in the plan to perform her actions.

Axiom 6 $\forall PE, S : PE \subseteq S \equiv$
$\forall_{A_i \in A_{PE}} PE_{A_i} \subseteq S \wedge \mathbf{SubPlanOf}(P_{PE_{A_i}}, P_{PE}, \mathbf{now_{PE}}) \wedge$
$S_{PE} = 0 \vee$
$\forall a_i : \mathbf{Uninstantiated}(a_i, PE_{A_i}) \supset$
$S \models \mathbf{Obliged}(A_i, a_i, A_{PE} - \{A_i\}, P_{PE})$

The single agent PES, PE_{A_1}, which corresponds to the MPES, PE, may be only part of the intentional structure for that agent. For example, the multi-agent plan might only include a particular action, a_i, whereas the agent who is to perform a_i might have an intentional structure including additional constraints on performance, or even extra actions which might *generate* a_i. In this case, the other agents will only be committed to the occurrence of the main action itself, not the particular way it might come about. Similarly, although A_1 might have intentions corresponding to the actions in some PE', of which PE_{A_1} is only a part, she will be obligated to the other agents of PE only for those a_i in $\mathbf{Actions}(P_{PE_{A_1}})$ and not the other sub-actions which comprise it in $P_{PE'}$.

Multi-agent PES's are a fairly fragile form of coordinated activity: any agent could repair or cancel her private PES, which could cancel the MPES. But this is as it should be: once one agent cancels her part, there is no longer a plan execution including that agent. However, the agent's obligations to perform her actions still persist, as do the commitments of the other agents to the occurrence of the agent's actions. To discharge her obligations, even after cancelling a private plan, an agent will either still have to perform the obliged actions, or receive some sort of permission from the other agents

(which might be automatic upon notifying them of the change in intention). If one agent simply cancels, the other agents will still be part of an MPES in which the occurrence of the agent's actions are part of the constraints. This will, in most cases, require an agent to communicate any changes to an individual plan execution which is part of a MPES. Also, if performance of an agent's actions is not evident to the other agents, this might also be a reason for communicating the successful performance.

7.6 Communication Recipes and Grounding Acts

We are now in a position to begin to define the grounding acts presented in Chapters 3 and 4 in terms of multi-agent plan executions.

7.6.1 The Abstract Communication Recipe: CR

Figure 7.1 is an abstract plan recipe for one agent (INITIATOR(CR)) communicating some content (CONTENT(CR)) to another agent (RESPONDER(CR)). This abstract recipe will be called CR (for Communication Recipe). Our claim is that successful execution of (a specialization of) this recipe will result in the (mutually assumed) mutual belief between the two agents that INITIATOR(CR) has communicated CONTENT(CR) to RESPONDER(CR). Agents engaged in conversation can be modelled as executing multi-agent plans which are specializations of this recipe. We will call any plan which has as its recipe a specialization of CR a *conversation plan.*

$$\textbf{Actions}(\text{CR}) = \{\text{present}_i \mid \text{present}_i = \text{Present}(\text{Agent1}_i, \text{Content}_i, \text{Recipient1}_i, \text{t1}_i)\} \cup \{\text{ack}_i \mid \text{ack}_i = \text{Ack}(\text{Agent2}_i, \text{Content}_i, \text{Recipient2}_i, \text{t2}_i)\}$$

$$\begin{array}{ll} \textbf{Constraints}(\text{CR}) = & \{ \\ \text{Temporal constraints} & \forall i(\text{Before}(\text{t1}_i, \text{t2}_i)), \\ \text{Agent constraints} & \forall i(\text{Agent1}_i = \text{Recipient2}_i = \text{INITIATOR(CR)}, \\ & \quad \text{Agent2}_i = \text{Recipient1}_i = \text{RESPONDER(CR)}), \\ \text{Object Constraints} & \text{CONTENT(CR)} = \cup_i \text{Content}_i\} \end{array}$$

Figure 7.1: Plan Recipe for Communication (Recipe CR)

The acts of presenting and acknowledging the content are broken into some indeterminate number of conceptual sub-acts, about at the granularity of the propositions in [Hobbs, 1985]. The only constraints on performance of these is that for a particular piece of the content, the presentation must come before the acknowledgement, and both must occur to achieve mutual belief. Using the formulation in section 7.4, we notice that any sub-recipe containing both presentations and acknowledgements of parts of the CONTENT in the proper order will itself be a communication recipe to present that sub-content. This notion of a communication recipe is modelled on the notion of *Contribution* proposed by Clark and Schaefer [1989]. The presentation phase will be realizations of the present_i acts, while the acceptance phase will realize the ack_i acts.

As an example, consider a recipe for Ag_1 to suggest a MoveEngine action to Ag_2. In that case CONTENT would be A = Suggest(MoveEngine(Ag_A, $Source_A$,$Dest_A$,$time_A$))[7] and we would have the distribution of abstract content shown in (53), each requiring a presentation by Ag_1 and an acknowledgement by Ag_2.

(53) $Content_1$=SpeechactType(A)=Suggest $Content_2$ = Pred(A) = MoveEngine
$Content_3$ = Agent(Pred(A)) = Ag_A $Content_4$=Source(Pred(A))=$Source_A$
$Content_5$ = Dest(Pred(A)) = $Dest_A$ $Content_6$ = time(Pred(A)) = $time_A$

Constraints on exactly what types of executions can present and acknowledge the above contents will be determined by conventions of the particular language used and the communicative contexts. For any given execution, some parts of the content will be expressed explicitly as part of the compositional conventional meaning of the utterance, and others will be presented implicitly by conventions of situated meaning and Gricean implicatures. As an example, uttering "Let's have Engine E3 move to Bath" explicitly provides the illocutionary force of a suggestion ($Content_1$), the agent ($Content_3$), and the Destination ($Content_5$), the other terms of the source and time of the action are either not provided or provided implicitly, depending on contextual-based inference – e.g., the source ($Content_4$) might be considered to be the current location of the engine. As another example, uttering "ok" (with proper intonation) after another's utterance will generally signal acknowledgement of all of the presented items which were part of the communication plan of that utterance.

When Ag_1 decides to make the suggestion, she forms a multi-agent plan, P_0, with recipe shown in Figure 7.2. This recipe is a specialization of CR, the communication recipe for the particular content which Ag_1 wants to communicate.

$$\begin{aligned}\textbf{Actions}(P_0) = \{\ & present_1 = Present(Ag_1, Content_1, Ag_2, t1_1), \ldots, \\ & present_6 = Present(Ag_1, Content_6, Ag_2, t1_6), \\ & ack_1 = Ack(Ag_2, Content_1, Ag_1, t2_1), \ldots, \\ & ack_6 = Ack(Ag_2, Content_6, Ag_1, t2_6)\ \} \\ \textbf{Constraints}(P_0) = \{\ & Before(t1_1, t2_1), \ldots, Before(t1_6, t2_6)\ \}\end{aligned}$$

Figure 7.2: Recipe for communication plan P_0

Ag_1 will also adopt P_1, her part of the multi-agent plan, as shown in Figure 7.3.

As part of adopting these plans, Ag_1 adopts the attitudes for her part of the multi-agent plan execution, shown in (54) as part of the resulting discourse situation, **DS**.[8]

[7]Cohen and Levesque [1990a] have suggested that illocutionary force recognition is unnecessary for communication, that speech acts are not primitives and need not be recognized explicitly, but are just retrospective labelings which might be heuristically useful. We maintain the latter point, that they are heuristically useful, while remaining neutral on the larger question of whether or not they are necessary. It is the case, for example, that certain syntactic, lexical, and intonational elements of utterances are good signals of ranges of illocutionary force. Note that the analysis given here in no way relies on the outcome of this question. We assume here that the illocutionary force is one parameter of the presentation, but it could just as easily be replaced with whatever the more basic formulation, in terms of beliefs and intentions, that such a speech act would turn out to be.

[8]**DS** is really an indexical expression which always refers to the current discourse situation.

$\mathbf{Actions}(P_1) = \{ present_1, \ldots, present_6 \}$
$\mathbf{Constraints}(P_1) = \{ Before(t1_1, t2_1), \ldots, Before(t1_6, t2_6),$
$Occurs(ack_1), \ldots, Occurs(ack_6) \}$

Figure 7.3: Recipe for plan P_1

If we give the name PE_1 to the initial PES for the execution of P_1, then this PES is a part of DS, (e.g., $PE_1 \subseteq$DS). The representation of PE_1 is shown in (55).

(54) $DS \models \mathbf{Intends}(Ag_1, present_1, P_0) \wedge \ldots \wedge \mathbf{Intends}(Ag_1, present_6, P_0) \wedge$
$\mathbf{Intends}(Ag_1, present_1, P_1) \wedge \ldots \wedge \mathbf{Intends}(Ag_1, present_6, P_1) \wedge$
$\mathbf{Committed}(Ag_1, Occurs(ack_1)) \wedge \ldots \wedge \mathbf{Committed}(Ag_1, Occurs(ack_6)) \wedge$
$\mathbf{Committed}(Ag_1, Before(t1_1, t2_1)) \wedge \ldots \wedge$
$\mathbf{Committed}(Ag_1, Before(t1_6, t2_6))$

(55) $PE_1 = [Ag_1, P_1, \{\}, \{\}, 1]$

Note that since Ag_2 does not yet have any intentions, there is no multi-agent plan execution situation with status 1 yet. However, Ag_1 still intends her actions to be part of a multi-agent communication plan, P_0, as well as her single-agent part of the plan, P_1. If Ag_1 now performs α_1, saying the words above (let's say in a context where the source and time of the action are apparent from context), this will update **DS** to now include PE_2, shown in (56).

(56) $PE_2 = \mathbf{Update}(\alpha_1, PE_1) = [Ag_1, P_1, \{\alpha_1\}, \{present_1 \rightsquigarrow \alpha_1, \ldots, present_6 \rightsquigarrow \alpha_1\}, 1]$

Now, however, Ag_2 will observe α_1, and realize that Ag_1 is executing a communication plan. If the communication is successful, Ag_2 will recognize that α_1 realizes the actions that Ag_1 intended, and Ag_2 will come to adopt the multi-agent plan P_0 as well. In addition, Ag_2 will adopt P_2, his part of P_0. Figure 7.4 shows the recipe for P_2, while (57) shows some of the updated attitudes that are now part of the discourse situation. Note that in (57), Ag_1 is committed to α_1 fulfilling the *present* actions, while Ag_2 is merely committed to the occurrence of these actions (although he believes that they have been realized by α_1).

$\mathbf{Actions}(P_2) = \{ ack_1, \ldots, ack_6 \}$
$\mathbf{Constraints}(P_2) = \{ Before(t1_1, t2_1), \ldots, Before(t1_6, t2_6),$
$Occurs(present_1), \ldots, Occurs(present_6) \}$

Figure 7.4: Recipe for plan P_2

(57) $DS \models \mathbf{Bel}(Ag_1, \alpha_1 \triangleright present_1) \wedge \ldots \wedge \mathbf{Bel}(Ag_1, \alpha_1 \triangleright present_6) \wedge$
$\mathbf{Committed}(Ag_1, \alpha_1 \triangleright present_1) \wedge \ldots \wedge \mathbf{Committed}(Ag_1, \alpha_1 \triangleright present_6) \wedge$

$$\begin{aligned}
&\mathbf{Committed}(\mathrm{Ag}_1, \mathrm{Occurs}(\mathrm{ack}_1)) \wedge \ldots \wedge \mathbf{Committed}(\mathrm{Ag}_1, \mathrm{Occurs}(\mathrm{ack}_6)) \wedge \\
&\mathbf{Bel}(\mathrm{Ag}_2, \alpha_1 \triangleright \mathrm{present}_1) \wedge \ldots \wedge \mathbf{Bel}(\mathrm{Ag}_2, \alpha_1 \triangleright \mathrm{present}_6) \wedge \\
&\mathbf{Intends}(\mathrm{Ag}_2, \mathrm{ack}_1, \mathrm{P}_0) \wedge \ldots \wedge \mathbf{Intends}(\mathrm{Ag}_2, \mathrm{ack}_6, \mathrm{P}_0) \wedge \\
&\mathbf{Intends}(\mathrm{Ag}_2, \mathrm{ack}_1, \mathrm{P}_2) \wedge \ldots \wedge \mathbf{Intends}(\mathrm{Ag}_2, \mathrm{ack}_6, \mathrm{P}_2) \wedge \\
&\mathbf{Committed}(\mathrm{Ag}_2, \mathrm{Occurs}(\mathrm{present}_1)) \wedge \ldots \wedge \\
&\mathbf{Committed}(\mathrm{Ag}_2, \mathrm{Occurs}(\mathrm{present}_6)) \wedge \\
&\mathbf{Obliged}(\mathrm{Ag}_2, \mathrm{ack}_1, \mathrm{Ag}_1, \mathrm{P}_0) \wedge \ldots \wedge \mathbf{Obliged}(\mathrm{Ag}_2, \mathrm{ack}_6, \mathrm{Ag}_1, \mathrm{P}_0)
\end{aligned}$$

The situation of Ag_2 executing his part of the multi-agent plan, PE_3, is shown in (58), and the complete multi-agent plan execution situation, PE_4, is shown in (59).

(58) $\mathrm{PE}_3 = [\mathrm{Ag}_2, \mathrm{P}_2, \{\}, \{\}, 1]$

(59) $\mathrm{PE}_4 = [\{\mathrm{Ag}_1, \mathrm{Ag}_2\}, \mathrm{P}_0, \{\alpha_1\}, \{\mathrm{present}_1 \rightsquigarrow \alpha_1, \ldots, \mathrm{present}_6 \rightsquigarrow \alpha_1\}, 1]$

Now, suppose Ag_2 performs α_2, attempting to acknowledge all the pieces of the content at once, by uttering "okay". The updated MPES is PE_5, shown in (60)a. The updated PES for Ag_2 is shown in (60)b, while (60)c shows the updated PES for Ag_1. Note that the PE_7 is the same as PE_2, since Ag_1 has taken no further action.

(60) a. $\mathrm{PE}_5 = \mathbf{Update}(\alpha_2, \mathrm{PE}_4) =$
$[\{\mathrm{Ag}_1, \mathrm{Ag}_2\}, \mathrm{P}_0, \{\alpha_1, \alpha_2\}, \{\mathrm{present}_1 \rightsquigarrow \alpha_1, \ldots, \mathrm{present}_6 \rightsquigarrow \alpha_1,$
$\mathrm{ack}_1 \rightsquigarrow \alpha_2, \ldots, \mathrm{ack}_6 \rightsquigarrow \alpha_2 \quad \}, 1]$

b. $\mathrm{PE}_6 = \mathbf{Update}(\alpha_2, \mathrm{PE}_3) = [\mathrm{Ag}_2, \mathrm{P}_2, \{\alpha_2\}, \{\mathrm{ack}_1 \rightsquigarrow \alpha_2, \ldots, \mathrm{ack}_6 \rightsquigarrow \alpha_2\}, 1]$

c. $\mathrm{PE}_7 = \mathbf{Update}(\alpha_2, \mathrm{PE}_2) = [\mathrm{Ag}_1, \mathrm{P}_1, \{\alpha_1\}, \{\mathrm{present}_1 \rightsquigarrow \alpha_1, \ldots, \mathrm{present}_6 \rightsquigarrow \alpha_1\}, 1]$

Some of the updated attitudes are shown in (61). According to the beliefs of both agents, the actions have all been performed and the constraints met, so they will each believe that the communication has been successful.

$$\begin{aligned}
(61)\ \mathrm{DS} \models\ &\mathbf{Bel}(\mathrm{Ag}_1, \alpha_1 \triangleright \mathrm{present}_1) \wedge \ldots \wedge \mathbf{Bel}(\mathrm{Ag}_1, \alpha_1 \triangleright \mathrm{present}_6) \wedge \\
&\mathbf{Bel}(\mathrm{Ag}_1, \alpha_2 \triangleright \mathrm{ack}_1) \wedge \ldots \wedge \mathbf{Bel}(\mathrm{Ag}_1, \alpha_2 \triangleright \mathrm{ack}_6) \wedge \\
&\mathbf{Committed}(\mathrm{Ag}_1, \alpha_1 \triangleright \mathrm{present}_1) \wedge \ldots \wedge \mathbf{Committed}(\mathrm{Ag}_1, \alpha_1 \triangleright \mathrm{present}_6) \wedge \\
&\mathbf{Committed}(\mathrm{Ag}_1, \mathrm{Occurs}(\mathrm{ack}_1)) \wedge \ldots \wedge \mathbf{Committed}(\mathrm{Ag}_1, \mathrm{Occurs}(\mathrm{ack}_6)) \wedge \\
&\mathbf{Bel}(\mathrm{Ag}_2, \alpha_1 \triangleright \mathrm{present}_1) \wedge \ldots \wedge \mathbf{Bel}(\mathrm{Ag}_2, \alpha_1 \triangleright \mathrm{present}_6) \wedge \\
&\mathbf{Bel}(\mathrm{Ag}_2, \alpha_2 \triangleright \mathrm{ack}_1) \wedge \ldots \wedge \mathbf{Bel}(\mathrm{Ag}_2, \alpha_2 \triangleright \mathrm{ack}_6) \wedge \\
&\mathbf{Committed}(\mathrm{Ag}_2, \alpha_2 \triangleright \mathrm{ack}_1) \wedge \ldots \wedge \mathbf{Committed}(\mathrm{Ag}_2, \alpha_2 \triangleright \mathrm{ack}_6) \wedge \\
&\mathbf{Committed}(\mathrm{Ag}_2, \mathrm{Occurs}(\mathrm{present}_1)) \wedge \ldots \wedge \mathbf{Committed}(\mathrm{Ag}_2, \mathrm{Occurs}(\mathrm{present}_6)
\end{aligned}$$

7.6.2 Defining Grounding Acts

Although in Section 7.4, above, we were noncommittal about whether plans had to have sub-plans for all of their sub-recipes, we will now require a domain axiom that Conversation Plans (e.g., those with a recipe which is a specialization of the conversation recipe, CR) have sub-plans corresponding to all sub-recipes which are also specializations of CR.

Axiom 7 $\forall P, t, R : \mathbf{Recipe}(P, t) \sqsubseteq CR \wedge R \subseteq \mathbf{Recipe}(P, t) \wedge R \sqsubseteq CR \supset$
$\exists P' : \mathbf{Recipe}(P', t) = R \wedge \mathbf{SubPlanOf}(P', P, t)$

The grounding acts introduced in Chapter 3 were defined there only operationally, in terms of the contexts in which they could appear and how they change the beliefs, obligations, and mutual beliefs of the participants. With the analysis here, it is now possible to define them precisely in terms of executions that are part of a PES with plan recipe a specialization of the abstract recipe, CR, given in Figure 7.1.

First we define two predicates on PESs: **Acknowledged** and **Unacknowledged.** **Unacknowledged**(PE) is true iff some part of CONTENT(**Recipe**(P_{PE})) has been presented, but not acknowledged. **Acknowledged**(PE) is true iff all parts of CONTENT(**Recipe**(P_{PE})) which have been presented have also been acknowledged. Neither would hold for a PE in which the recipe were not a specialization of CR. Formally:

Definition 26 **Unacknowledged**(PE) iff $\mathbf{Recipe}(P_{PE}) \sqsubseteq CR \wedge$
$\exists i : \mathbf{Instantiated}(present_i, PE) \wedge \mathbf{Uninstantiated}(ack_i, PE)$

Definition 27 **Acknowledged**(PE)) iff
$\mathbf{Recipe}(P_{PE}) \sqsubseteq CR \wedge \forall i : \mathbf{Instantiated}(present_i, PE) \supset \mathbf{Instantiated}(ack_i, PE)$

Thus, for the example in the previous section, both **Unacknowledged**(PE_4) and **Acknowledged**(PE_5) would hold.

Given that α, a communicative action, is performed in some discourse situation **DS**, then the conditions for α realizing one of the grounding acts are as follows.

α realizes CONTINUE(DU_n), a continuation of a Discourse unit, DU_n, iff α continues a communication plan, and there is a presentation in that plan that has not been acknowledged. Formally:

$DS \models \alpha \triangleright \text{CONTINUE}(DU_n)$ iff
$\exists PE : \mathbf{Recipe}(P_{PE}) \sqsubseteq CR \wedge PE \subseteq DS \wedge \mathbf{Unacknowledged}(PE) \wedge$
$\mathbf{CONTINUES}(\alpha, PE) \wedge \alpha \triangleright \{a_i \mid \forall a_i : a_i = present_j \wedge \mathbf{Uninstantiated}(a_i, PE)\}$

α realizes INITIATE(DU_n) iff α begins a communication plan, and if it continues any other communication plan, then that plan must have no unacknowledged presentations. Formally:

$DS \models \alpha \triangleright \text{INITIATE}(DU_n)$ iff
$\exists PE : \textbf{Recipe}(P_{PE}) \sqsubseteq CR \wedge PE \subseteq DS \wedge \neg[DS \models \alpha \triangleright \text{CONTINUE}(DU_n)] \wedge$
$\textbf{BEGINS}(\alpha, PE) \wedge \alpha \triangleright \{a_i \mid \forall a_i : a_i = present_j \wedge \textbf{Uninstantiated}(a_i, PE)\}$

α realizes ACKNOWLEDGE(DU_n) iff α completes a communication plan. Formally:

$DS \models \alpha \triangleright \text{ACKNOWLEDGE}(DU_n)$ iff
$\exists PE : \textbf{Recipe}(P_{PE}) \sqsubseteq CR \wedge PE \subseteq DS \wedge \textbf{COMPLETES}(\alpha, PE) \wedge$
$\alpha \triangleright \{a_i \mid \forall a_i : a_i = ack_j \wedge \textbf{Instantiated}(present_j, PE)\}$

In the above example, the discourse situation, **DS**, will support the following grounding act interpretations: $\alpha_1 \triangleright \text{INITIATE}(DU_1)$ and $\alpha_2 \triangleright \text{ACKNOWLEDGE}(DU_1)$.

A slightly more complex example, illustrating the differences between INITIATE and CONTINUE is shown in Figure 7.5, an extract from a collected conversation in the TRAINS domain[9].

Act	UU#	Speaker: Utterance
$init_1$	1.4	M: first thing I'd like you to do
$cont_1(1.4)$	1.5	: is send engine E2 off with a boxcar to Corning to pick up oranges
$cont_1(1.5)$	1.6	: uh as soon as possible
ack_1	2.1	S: okay
$init_2$	3.1	M: and while it's there it should pick up the tanker
ack_2	4.1	S: okay

Figure 7.5: Conversation Fragment with Grounding Acts

Here, the entire fragment can be seen as the execution of one communication plan for M to communicate to S his desires concerning a domain plan. (62) shows a breakdown of some of the content to be expressed in a communication recipe that might have led to this dialogue, although not in as much detail as given in (53).

(62) $Content_1 = SpeechactType(B) = Suggest$ $Content_2 = Act_1(B) = MoveEngine$
$Content_3 = Engine(Act_1(B)) = E2$ $Content_4 = Coupled(E2, \lambda bc, time(Act_1(B))$
$Content_5 = Purpose(Act_1(B)) = Act_2(B) = LoadOranges$
$Content_6 = Time(Act_1(B)) = ASAP$ $Content_7 = Act_3(B) = Couple$
$Content_8 = After(Time(Act_1(B)), time(Act_3(B)))$

Initially, M will adopt intentions and commitments which are in accord with executing the multi-agent plan P_3, shown in Figure 7.6, and his part of that plan, P_4, shown in Figure 7.7.

The first utterance by M (UU1.4) realizes $pres_1$ and thus $\mathbf{DS} \models \mathbf{Begin}(P_3)$. Since S has no idea at this point of the whole extent of P_3, there is still no multi-agent plan execution of P_3 with status 1. However, the agents are now engaged in a multi-agent

[9]Taken from Dialogue 91-3.2 in [Gross *et al.*, 1993].

$$\textbf{Actions}(P_3) = \{\ pres_1 = \text{Present}(M, \text{Content}_1, S, t1_1), \ldots,$$
$$pres_8 = \text{Present}(M, \text{Content}_8, S, t1_8),$$
$$ack_1 = \text{Ack}(S, \text{Content}_1, M, t2_1), \ldots,$$
$$ack_8 = \text{Ack}(S, \text{Content}_8, M, t2_8)\ \}$$
$$\textbf{Constraints}(P_3) = \{\ \text{Before}(t1_1, t2_1), \ldots, \text{Before}(t1_8, t2_8)\ \}$$

Figure 7.6: Recipe for multi-agent communication plan P_3

$$\textbf{Actions}(P_4) = \{\ pres_1, \ldots, pres_8\}$$
$$\textbf{Constraints}(P_4) = \{\ \text{Before}(t1_1, t2_1), \ldots, \text{Before}(t1_8, t2_8),$$
$$\text{Occurs}(ack_1), \ldots, \text{Occurs}(ack_8)\ \}$$

Figure 7.7: Recipe for plan P_4

plan execution of a plan (we'll call it P_5) with recipe shown in Figure 7.8, which, by Axiom 7, is a sub-plan of P_3. The PES's shown in (63) are all part of the discourse situation resulting after UU1.4. UU1.4 is thus an INITIATE of DU_1, since it is the first presentation action in the execution of communication plan P_5.

$$\textbf{Actions}(P_5) = \{\ pres_1, ack_1\}$$
$$\textbf{Constraints}(P_5) = \{\ \text{Before}(t1_1, t2_1)\ \}$$

Figure 7.8: Recipe for communication plan P_5

(63) a. $PE_7 = [\{M, S\}, P_3, \{UU1.4\}, \{pres_1 \rightsquigarrow UU1.4\}, 0]$

b. $PE_8 = [M, P_4, \{UU1.4\}, \{pres_1 \rightsquigarrow UU1.4\}, 1]$

c. $PE_9 = [\{M, S\}, P_5, \{UU1.4\}, \{pres_1 \rightsquigarrow UU1.4\}, 1]$

Utterance 1.5 realizes $pres_2$, $pres_3$, $pres_4$, and $pres_5$. It continues M's top-level plan (i.e., $\mathbf{DS} \models \mathbf{Continue}(P_4)$), and S notices that it is also meant as part of a communication plan including UU1.4. Although utterance 1.5 begins the execution of a plan with recipe shown in Figure 7.9, it also continues execution of a plan with recipe shown in Figure 7.10, and thus this utterance is a CONTINUE rather than an INITIATE.

Some of the PES's which are part of the updated discourse situation, as well as some of the relationships of UU1.5 to the various plans are shown in (64).

(64) a. $PE_{10} = \mathbf{Update}(UU1.5, PE_8) =$
$[M, P_4, \{UU1.4, UU1.5\}, \{pres_1 \rightsquigarrow UU1.4, pres_2 \rightsquigarrow UU1.5, pres_3 \rightsquigarrow UU1.5,$
$pres_4 \rightsquigarrow UU1.5, pres_5 \rightsquigarrow UU1.5\ \},1]$

b. $PE_{11} = [\{M, S\}, P_6, \{UU1.5\}, \{pres_2 \rightsquigarrow UU1.5, \ldots, pres_5 \rightsquigarrow UU1.5\}, 1]$

Actions(P_6) = { $pres_2, \ldots, pres_5, ack_2, \ldots, ack_5$ }
Constraints(P_6) = { $Before(t1_2, t2_2), \ldots, Before(t1_5, t2_5)$ }

Figure 7.9: Recipe for communication plan P_6

Actions(P_7) = { $pres_1, \ldots, pres_5, ack_1, \ldots, ack_5$ }
Constraints(P_7) = { $Before(t1_1, t2_1), \ldots, Before(t1_5, t2_5)$ }

Figure 7.10: Recipe for communication plan P_7

c. $PE_{12} = [\{M, S\}, P_7, \{UU1.4, UU1.5\}, \{pres_1 \rightsquigarrow UU1.4, \ldots, pres_5 \rightsquigarrow UU1.5\}, 1]$

d. $DS \models UU1.5 \triangleright$ **Begin**(P_6) $\wedge$ $UU1.5 \triangleright$ **Continue**(P_7) $\wedge$ $UU1.5 \triangleright$ **Continue**(P_3)

UU1.6 realizes $pres_6$, and so is also a CONTINUE of DU_1. UU2.1 realizes ack_1, ..., ack_6. It completes several of the sub-plans of P_3, and is thus an ACKNOWLEDGE of DU_1. Figure 7.11 shows the plan for this entire DU, while (65) describes some features of the updated discourse situation.

Actions(P_8) = { $pres_1, \ldots, pres_6, ack_1, \ldots, ack_6$ }
Constraints(P_8) = { $Before(t1_1, t2_1), \ldots, Before(t1_6, t2_6)$ }

Figure 7.11: Recipe for communication plan P_8

(65) a. PE_{13} = **Update**(UU2.1, **Update**(UU1.5, PE_{10})) =
$[M, P_4, \{UU1.4, UU1.5, UU1.6\}, \{pres_1 \rightsquigarrow UU1.4, pres_2 \rightsquigarrow UU1.5, pres_3 \rightsquigarrow UU1.5,$
$pres_4 \rightsquigarrow UU1.5, pres_5 \rightsquigarrow UU1.5, pres_6 \rightsquigarrow UU1.6\}, 1]$

b. $PE_{14} = [\{M, S\}, P_8, \{UU1.4, UU1.5, UU1.6, UU2.1\},$
$\{pres_1 \rightsquigarrow UU1.4, \ldots, pres_6 \rightsquigarrow UU1.6, ack_1 \rightsquigarrow UU2.1, \ldots, ack_6 \rightsquigarrow UU2.1\}, 1]$

c. $DS \models UU2.1 \triangleright$ **Complete**(P_8)

UU3.1 realizes $pres_7$ and $pres_8$, and so continues M's highest level communication plan P_4, as his part of P_4. However, since there are no unacknowledged PES's as part of the discourse situation, this is not a CONTINUE. Thus, it meets the conditions to be an INITIATE of a new DU (DU_2), as it begins execution of the plan in Figure 7.12.

UU4.1 realizes ack_7 and ack_8, and thus completes P_9, as well as completing the original multi-agent plan P_3. This example illustrates that sequential interaction is more important for grounding act classification than any original plans. If utterance 2.1 did not precede 3.1, then 3.1 would have been a CONTINUE. Similarly, if 2.1 had preceded 1.6, then 1.6 would have been an INITIATE.

Going back to the definitions of the other grounding acts, α realizes REPAIR(DU_n) iff α repairs a communication plan. Formally,

Actions(P_9) = { $pres_7$, $pres_8$, ack_7, ack_8 }
Constraints(P_9) = { Before($t1_7$, $t2_7$), Before($t1_8$, $t2_8$) }

Figure 7.12: Recipe for communication plan P_9

DS $\models$ $\alpha \triangleright$REPAIR(DU_n) iff
$\exists$PE : **Recipe**(P_{PE})$\sqsubseteq$CR $\wedge$ PE$\subseteq$DS $\wedge$ DS$\models$ $\alpha \triangleright$**Repair**(P_{PE})

In practice, only **ERepairs** will be detectable, barring repair markers such as disfluencies in speech. If an initiator changes her mind about some portion of what to communicate before that part has been presented (a **PRepair** from a plan execution point of view), it may well be seen as a CONTINUE by the responder. Consider S's point of view after UU1.5, in the above example. As far as S knows, M's original plan may have been P_7. S has no way of knowing if UU1.6 just continues the original plan (P_3, or even P_8), or is a **PRepair** adding on a new action. Likewise, if an initiating agent performs a **PRepair** to remove an unpresented part of the content, this will generally be undetectable. In cases in which a REPAIR is ambiguous with a CONTINUE, generally the CONTINUE will be chosen as the preferred act interpretation.

α realizes CANCEL(DU_n) iff α cancels a communication plan. Formally,

DS $\models$ $\alpha \triangleright$CANCEL(DU_n) iff
$\exists$PE : **Recipe**(P_{PE})$\sqsubseteq$CR $\wedge$ PE$\subseteq$DS $\wedge$ **CANCELS**(α, PE)

As with REPAIR, only cases in which some presentation has already been made will be detectable as CANCELs.

In order to fully formalize the remaining acts, REQUEST-REPAIR, and REQUEST-ACKNOWLEDGE, we need to formalize requests in general, which, as mentioned in previous chapters, we take as placing a *discourse obligation* on the requested party to respond. These acts will not generally change the plan execution itself, but will add obligations to the discourse situation of which the plan execution is a part. Also, in this context, a REQUEST-ACKNOWLEDGE signals that the presentation part of some plan with a sub-recipe of CR is deemed complete, and the acknowledgement part should be forthcoming. A REQUEST-REPAIR signals that there is some problem with some of the presentations of the current plan with with a sub-recipe of CR are errors and must be repaired by the requested party.

7.6.3 Discourse Units and Conversation Plans

We are now in a position to characterize the participation of agents in constructing discourse units as execution of conversation plans. Given the above definitions of the grounding acts, a completed discourse unit can be seen as the execution of a maximal

conversation plan for which all of the presentation acts together occur before any of the acknowledgements.[10]

Agents need not have plans to produce the *specific* DU structure. For example, the precise relative timing of the individual presentations and acknowledgements may remain unspecified until actual execution. The participants need not even predetermine the sub-plan which will end up comprising a DU. The DU structure will result from the placement of acknowledgements relative to other presentations, which may occur at any of a number of positions, as the example in Figure 7.5 is intended to show.

7.6.4 Intermediate Grounding Acts

The theory in this chapter and the abstract conversation recipe CR provide a good foundation for taking up the issues raised in Section 3.5, particularly *partial acknowledgements*, in which an utterance seems to ground some material, while repairing other material. This kind of behavior can be naturally explained using the conversation recipe CR. One of these intermediate acts would be performing the ack acts for some bits of the content, while performing repairs or repair requests, signaling problems with the presentations of others.

An agent who merely tracks (a hypothesis about) the current conversation plans will be able, at any given stage in the conversation, to determine for each bit of content assumed to be part of the recipe whether it has been presented and/or acknowledged. Of course, given the vagueness and context and inference dependence of NL communication, it will not always be possible to tell, for a given utterance, just which presentations and acknowledgements the other agent will decide have been performed. Whenever this uncertainty becomes unmanageable (surpassing the Clark and Schaefer's *grounding criterion*) an agent can decide to repair.

[10] Actually, this condition is slightly too strict, because a repair may occur after an acknowledgement, undoing one of the presentation acts which might then be reperformed *after* some other presentations have been acknowledged. Still, all of the *original* presentations must be performed before any of the acknowledgements in order for the plan execution to form a DU.

8 Conclusions and Future Research

8.1 Summary

This thesis has considered the phenomena of grounding in natural language conversation from several directions. Chapter 1 introduced the problem and contrasted it with the related but much simpler problem of grounding in computer communication protocols. Chapter 2 considered previous approaches to the problem, as well as models of action and mental attitudes used in constructing a conversational agent. Chapter 3 presented the basic grounding protocol, comparing it both to a more complex context-free model and pointing out the problem areas that remain. A cognitive model was also presented which related the performance of actions described by the protocol to the mental states of conversing agents. Chapter 4 described the relation of this protocol and the *grounding acts* which comprise it to a more general theory of action in conversation, including traditional sentential level illocutionary acts (which are now modelled as multi-agent action, achieved only when grounded), turn-taking acts, and argumentation acts composed of multiple speech acts coordinated into higher-level communicative plans. Chapter 5 described an agent-based model of dialogue management and showed how grounding is just one of the functions to which such a system must attend. Chapter 6 described an implementation of the ideas of the previous chapters within the TRAINS-93 system, proving that such a framework *can* be used, at least for simple task oriented dialogues, to engage naturally in conversation. Chapter 7 steps back from the grounding protocol and describes the foundations for a theory of plan execution in which sub-actions and repairs can be given precise formulations. This theory is then used both to model the grounding process and to show how the possible extensions discussed in Chapter 3 also fit naturally into such a theory.

8.2 Future Work

Although we hope that the present work has shed some light on how to account for grounding within the framework of modelling a computational agent, there is still much that can be done. This section briefly explores some of the steps that might be taken in the future to improve and extend the present work.

8.2.1 Refining The Conversation Act Taxonomy

Although the speech act taxonomy presented in Chapter 4 was based on examination of a corpus of conversation, one thing that needs to be done is to go back and see how well this taxonomy really covers the corpus. An attempt should be made to answer the following questions: what percentage of acts can be classified reliably? How many utterances don't seem to fit the scheme? Does classifying the acts lead to acceptable assumptions about the intentions of the participants? While it is hard to quantify the acceptability of a classification scheme since there is no direct access to the mental states of the participants, examinations can still lead to some basis for comparison with other proposed taxonomies. Some constraints on an acceptable classification scheme are:

- Does it cover an acceptably large subset of the utterances to a sufficient degree?
- Are act types which seem to human analysts to be close to each other (e.g., those for which it is hard for people to distinguish which of the types a particular utterance is) shown to be close in the taxonomy? A hierarchical structure might help here so that ambiguities can be concisely represented and reasoned about.
- Is it possible to use the classification of acts towards recognizing the intentions of the speaker, and determining what to do next?
- What are tests which can distinguish one act from another? These tests should be based only on information available to an observer of the act. It should include syntactic, prosodic, and contextual information, but not be based on some private knowledge of the speaker's mental state which is not deducible from prior context.

The present Conversation Act taxonomy accomplishes some of these goals, yet there is still much room for improvement. Study of the argumentation level is still in its very early stages. Although there is currently much work on discourse relations (e.g., [Rambow, 1993]) there is still little consensus on which relations are necessary and how they can best be used. There is still even much work to be done on specifying precisely the constraints on occurrence, effects, and principles for recognition even of the core speech acts.

8.2.2 Utterance Unit Size

Two important considerations are that of the proper granularity of grounding installments, and how much of the change to the (potential) common ground is mediated directly by grounding acts. Taking up the latter point first, not all changes to the common ground or proposed common ground are arrived at using the mechanisms described in Section 3.3. On a broader scale, some of the content which is added through the regular grounding process will have effects beyond the adoption of new (mutual) beliefs. For example, *clarification* argumentation acts may actually revise previous beliefs which had been added through the grounding process. Also, renunciations of previously grounded suggestions will have the same long-term effect as cancelled material. Similarly, it may be advantageous to treat the low-level production of material as separate

from the grounding process. Detecting simple speech repairs [Levelt, 1983] such as the repetition or replacement of words within an intonation phrase is probably a separate process from grounding, e.g., Heeman and Allen [1994] claim that most speech local repairs can be handled using only local clues within an utterance unit.

As illustrated in Chapter 7, there are many types of *begins*, *continues*, *repairs*, *cancels*, of which the grounding acts discussed in Chapter 3 are just those concerned with grounding and DUs. The key issue, for repairs in particular, is to determine which repairs are to be treated at the Grounding level, using mechanisms like those described in this thesis, and which are other repairs (e.g., speech repairs), which should be handled using other mechanisms.

In the present work, it has been assumed that there is a level of dialogue structure between the word and the turn, called an *Utterance Unit.* While this assumption is fairly common, it is still open to question what the main considerations should be in determining utterance unit boundaries. Some of the candidates include syntactic phrase completion, semantic or speech act completion, and various prosodic features, including *boundary tones* [Pierrehumbert, 1980] and pause length. If the utterance unit level is the proper one in which to consider grounding installments, further work specifying the nature of utterance units and the detection of UU boundaries will ease the task of integrating a theory of grounding into the totality of dialogue processing.

8.2.3 Degrees of Belief

The model of belief assumed here is a straightforward all or nothing model – something is either believed or it isn't. This kind of model is easily represented using belief spaces or modal logic, but is insufficient for some purposes. Certain phenomena involved with grounding (e.g. [Clark and Schaefer, 1989]'s *Grounding Criterion* and *Strength of Evidence Principle*) seem to require a graded model of belief, where different strengths of belief of understanding are needed for different purposes. If we adopted such a scheme, we might be able to separate out different types of acknowledgement and explain multiple acknowledgements.

There is some current work which might be helpful here. [Bunt, 1989] uses a *Suspects* attitude as intermediate between *Belief* and lack of belief to model the conditions on check acts. [Lambert, 1993] uses a three-level model, including *certain belief*, *strong belief*, and *weak belief.* [Walker, 1993] uses a limited attention model of belief which also might be useful in modelling explicit belief.

8.2.4 Speech acts as part of general account of multi-agent interaction

While speech is its own modality with several features which are distinctive from other types of action, there is still a significant overlap with other kinds of action. For instance, a request can be to perform another speech act or to perform some physical action. Speech acts are often made to help satisfy domain goals in similar ways to the way physical actions are made. In order to capture some of the regularities of conversation planning, it is necessary to say how it fits in to a larger account of planning in a multi

agent domain. Cohen & Levesque give the beginnings of such an account, but not in a form which is useful for an agent involved in planning and acting in the world. Chapter 7 is a first step on this path, but much still remains to do, including demonstrating that the framework is useful for on-line plan related inference.

Bibliography

[Allen, 1983a] James F. Allen, "Maintaining Knowledge About Temporal Intervals," *Communications of the ACM*, 26(11):832–843, November 1983.

[Allen, 1983b] James [F.] Allen, "Recognizing Intentions From Natural Language Utterances," In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse*. MIT Press, 1983.

[Allen, 1991] James F. Allen, "Time and Planning," In R. Pelavin J. Allen, H. Kautz and J. Tenenberg, editors, *Reasoning About Plans*. Morgan Kaufmann, 1991.

[Allen and Miller, 1991] James F. Allen and B. W. Miller, "The RHET System: A sequence of Self-Guided Tutorials," Technical Report 325, University of Rochester - Computer Science, July 1991.

[Allen and Perrault, 1980] James F. Allen and C. Raymond Perrault, "Analyzing Intention in Utterances," *Artificial Intelligence*, 15(3):143–178, 1980.

[Allen and Schubert, 1991] James F. Allen and Lenhart K. Schubert, "The TRAINS Project," TRAINS Technical Note 91-1, Computer Science Dept. University of Rochester, 1991.

[Appelt and Konolige, 1988] Douglas Appelt and Kurt Konolige, "A Nonmonotonic Logic for Reasoning about Speech Acts and Belief Revision," In *Proceedings of Second International Workshop on Non-Monotonic Reasoning*, pages 164–175, 1988.

[Austin, 1962] J. A. Austin, *How to Do Things with Words*, Harvard University Press, 1962.

[Bach and Harnish, 1979] K. Bach and R. M. Harnish, *Linguistic Communication and Speech Acts*, The MIT Press, 1979.

[Balkanski, 1990] Cecile T. Balkanski, "Modelling Act-Type Relations In Collaborative Activity," Technical Report 23-90, Harvard University Center for Research in Computing Technology, 1990.

[Barwise, 1989] Jon Barwise, *The Situation in Logic*, chapter 9: On the Model Theory of Common Knowledge, CSLI Lecture Notes: Number 17. Center for The Study of Language and Information, 1989.

[Beun, 1989] Robbert-Jan Beun, *The Recognition of Declarative Questions in Information Dialogues*, PhD thesis, Katholieke Universiteit Brabant, 1989.

[Bratman, 1987] Michael E. Bratman, *Intention, Plans, and Practical Reason*, Harvard University Press, 1987.

[Bratman, 1990] Michael E. Bratman, "What Is Intention?," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Bratman *et al.*, 1988] Michael E. Bratman, David J. Israel, and Martha E. Pollack, "Plans and Resource-Bounded Practical Reasoning," Technical Report TR425R, SRI International, September 1988, Appears in Computational Intelligence, Vol. 4, No. 4, 1988.

[Brennan, 1990] Susan Brennan, *Seeking and Providing Evidence for Mutual Understanding*, PhD thesis, Stanford University, 1990.

[Bruce, 1975] Bertram C. Bruce, "Generation as a Social Action," In *Theoretical Issues in Natural Language Processing-1*, pages 64–67, 1975, Also appears in [?], pp. 419-422.

[Bruce and Newman, 1978] Bertram C. Bruce and Denis Newman, "Interacting Plans," *Cognitive Science*, 2:195–233, 1978.

[Bunt, 1989] H. C. Bunt, "Information Dialogues as Communicative Action in Relation to Partner Modelling and Information Processing," In M.M Taylor, F. Neel, and D. G. Bouwhuis, editors, *The Structure of Multimodal Dialogue*. Elsevier Science Publishers B.V., 1989.

[Cawsey, 1990] Alison Cawsey, "A Computational Model of Explanatory Discourse," In Paul Luff, Nigel Gilbert, and David Frohlich, editors, *Computers and Conversation*. Academic Press, 1990.

[Cawsey, 1991] Alison Cawsey, "A Belief Revision Model of Repair Sequences in Dialogue," In Ernesto Costa, editor, *New Directions in Intelligent Tutoring Systems*. Springer-Verlag, forthcoming, 1991.

[Chapman, 1987] David Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence*, 32:333–377, 1987, also appears in [?].

[Clark, 1992] Herbert H. Clark, *Arenas of Language Use*, University of Chicago Press, 1992.

[Clark and Brennan, 1990] Herbert H. Clark and Susan E. Brennan, "Grounding in Communication," In L. B. Resnick, J. Levine, and S. D. Behrend, editors, *Perspectives on Socially Shared Cognition*. APA, 1990.

[Clark and Carlson, 1982] Herbert H. Clark and Thomas B. Carlson, "Critic's Beliefs about Hearers' Beliefs: A rejoinder to Johnson-Laird, Sperber, and Wilks," In N. V. Smith, editor, *Mutual Knowledge*. Academic Press, 1982.

[Clark and Marshall, 1981] Herbert H. Clark and Catherine R. Marshall, "Definite Reference and Mutual Knowledge," In Aravind K. Joshi, Bonnie L. Webber, and Ivan A. Sag, editors, *Elements of Discourse Understanding.* Cambridge University Press, 1981, also appears as Chapter 1 in [Clark, 1992].

[Clark and Schaefer, 1989] Herbert H. Clark and Edward F. Schaefer, "Contributing to Discourse," *Cognitive Science*, 13:259 - 94, 1989, also appears as Chapter 5 in [Clark, 1992].

[Clark and Wilkes-Gibbs, 1986] Herbert H. Clark and Deanna Wilkes-Gibbs, "Referring as a Collaborative Process," *Cognition*, 22, 1986, also appears as Chapter 4 in [Clark, 1992].

[Cohen, 1978] Phillip R. Cohen, *On Knowing What to Say: Planning Speech Acts*, PhD thesis, University of Toronto, 1978, Reproduced as TR 118 Department of Computer Science, University of Toronto.

[Cohen and Levesque, 1990a] Phillip R. Cohen and Hector J. Levesque, "Performatives in a Rationally Based Speech Act Theory," In *Proceedings ACL-90*, pages 79–88, 1990.

[Cohen and Levesque, 1990b] Phillip R. Cohen and Hector J. Levesque, "Persistence, Intention, and Commitment," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[Cohen and Levesque, 1990c] Phillip R. Cohen and Hector J. Levesque, "Rational Interaction as the Basis for Communication," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[Cohen and Levesque, 1991a] Phillip R. Cohen and Hector J. Levesque, "Confirmations and Joint Action," In *Proceedings IJCAI-91*, pages 951–957, 1991.

[Cohen and Levesque, 1991b] Phillip R. Cohen and Hector J. Levesque, "Teamwork," *Nous*, 35, 1991.

[Cohen and Perrault, 1979] Phillip R. Cohen and C. R. Perrault, "Elements of a Plan-Based Theory of Speech Acts," *Cognitive Science*, 3(3):177–212, 1979.

[Conte and Castelfranchi, 1993] Rosaria Conte and Cristiano Castelfranchi, "Norms as mental objects. From normative beliefs to normative goals," In *Working Notes AAAI Spring Symposium on Reasoning about Mental States: Formal Theories and Applications.*, pages 40–47, March 1993.

[Coulthard *et al.*, 1981] M. Coulthard, M. Montgomery, and D. Brazil, "Developing a Description of Spoken Discourse," In M. Coulthard and M. Montgomery, editors, *Studies in Discourse Analysis*, pages 1–50. Routledge & Kegan Paul, 1981.

[Demazeau and Muller, 1990] Y. Demazeau and J. P. Muller, editors, *Decentralized A.I.*, Elsevier Science Publishers B. V., 1990.

[Demazeau and Muller, 1991] Y. Demazeau and J. P. Muller, editors, *Decentralized A.I. 2*, Elsevier Science Publishers B. V., 1991.

[Devanbu and Litman, 1991] Premkumar T. Devanbu and Diane J. Litman, "Plan-based terminological reasoning," In James Allen, Richard Fikes, and Eric Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 128–138, Cambridge, MA, May 1991.

[Devlin, 1991] Keith Devlin, *Logic and Information*, Cambridge University Press, 1991.

[Duncan and Niederehe, 1974] Starkey Duncan, Jr. and George Niederehe, "On Signalling That It's Your Turn to Speak," *Journal of Experimental Social Psychology*, 10:234–47, 1974.

[Ferguson, 1992] George Ferguson, "Explicit Representation of Events, Actions, and Plans for assumption-Based Plan Reasoning," Technical Report 428, Computer Science Dept. University of Rochester, June 1992.

[Ferguson, 1994] George Ferguson, "Domain Plan Reasoning in TRAINS-93," Trains technical note, Computer Science Dept. University of Rochester, forthcoming, 1994.

[Ferguson and Allen, 1994] George Ferguson and James F. Allen, "Arguing about Plans: Plan Representation and Reasoning for Mixed-Initiative Planning," In *Proceedings of the Second International Conference on AI Planning Systems (AIPS-94)*, Chicago, IL, 15–17 June 1994.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, 2:189–208, 1971, Also appears in [?].

[Ford and Thompson, 1991] Cecilia Ford and Sandra Thompson, "On Projectability in conversation: Grammar, Intonation, and Semantics," presented at the Second International Cognitive Linguistics Association Conference, August, 1991.

[Frohlich and Luff, 1990] David Frohlich and Paul Luff, "Applying the Technology of Conversation to the Technology for Conversation," In Paul Luff, Nigel Gilbert, and David Frohlich, editors, *Computers and Conversation*. Academic Press, 1990.

[Galliers, 1990] Julia Rose Galliers, "Belief Revision and A Theory of Communication," Technical Report 193, Cambridge University Computer Laboratory, 1990.

[Goldman, 1970] Alvin I. Goldman, *A Theory of Human Action*, Prentice Hall Inc., 1970.

[Grice, 1957] H. P. Grice, "Meaning," *Philosophical Review*, 66:377–88, 1957.

[Gross *et al.*, 1993] Derek Gross, James Allen, and David Traum, "The TRAINS 91 Dialogues," TRAINS Technical Note 92-1, Computer Science Dept. University of Rochester, July 1993.

[Grosz and Kraus, 1993] Barbara [J.] Grosz and Sarit Kraus, "Collaborative Plans for Group Activities," In *Proceedings IJCAI-93*, pages 367–373, 1993.

[Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner, "Attention, Intention, and the Structure of Discourse," *CL*, 12(3):175–204, 1986.

[Grosz and Sidner, 1990] Barbara J. Grosz and Candace L. Sidner, "Plans for Discourse," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Halliday, 1961] M. A. K. Halliday, "Categories of the Theory of Grammar," *Word*, 17:241–92, 1961.

[Halpern and Moses, 1990] J. Y. Halpern and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," *Journal of the ACM*, 37(3):549–587, 1990.

[Harman, 1977] Gilbert Harman, "Review of *Linguistic Behaviour* by Jonathan Bennett," *Language*, 53:417–424, 1977.

[Heeman and Allen, 1994] Peter Heeman and James Allen, "Detecting and Correcting Speech Repairs," In *Proceedings ACL-94*, pages 295–302, Las Cruces, New Mexico, June 1994.

[Heritage, 1984] John Heritage, "A change-of-state token and aspects of its sequential placement," In J. M. Atkinson and J. Heritage, editors, *Structures of Social Action*. Cambridge University Press, 1984.

[Hinkelman, 1990] Elizabeth A. Hinkelman, *Linguistic and Pragmatic Constraints on Utterance Interpretation*, PhD thesis, University of Rochester, 1990.

[Hinkelman and Allen, 1989] Elizabeth A. Hinkelman and James F. Allen, "Two Constraints on Speech Act Ambiguity," In *Proceedings ACL-89*, pages 212–219, 1989.

[Hintikka, 1962] Jaakko Hintikka, *Knowledge and belief; an introduction to the logic of the two notions*, Cornell University Press, 1962.

[Hintikka, 1971] Jaakko Hintikka, "Semantics and Propositional Attitudes," In Leonard Linsky, editor, *Reference and Modality*, chapter X, pages 145–167. Oxford University Press, 1971.

[Hobbs, 1985] Jerry Hobbs, "Ontological Promiscuity," In *Proceedings ACL-85*, pages 61–69, 1985.

[Hobbs and Evans, 1979] Jerry Hobbs and David Evans, "Conversation as Planned Behavior," Technical Report 203, SRI International, December 1979.

[Hwang and Schubert, 1993] C. H. Hwang and L. K. Schubert, "Episodic Logic: A situational logic for natural language processing," In *Situation Theory and its Applications, V. 3*, CSLI, Stanford, CA, 1993.

[Johnson-Laird, 1982] P. N. Johnson-Laird, "Mutual ignorance: Comments on Clark and Carlson's Paper," In N. V. Smith, editor, *Mutual Knowledge*. Academic Press, 1982.

[Konolige, 1988] Kurt Konolige, "Hierarchic autoepistemic theories for nonmonotonic reasoning: preliminary report," In *Proceedings of Second International Workshop on Non-monotonic Reasoning*, pages 42–59, 1988.

[Konolige and Pollack, 1993] Kurt Konolige and Martha E. Pollack, "A Representationalist Theory of Intention," In *Proceedings IJCAI-93*, 1993.

[Lambert, 1993] Lynn Lambert, *Recognizing Complex Discourse Acts: A Tripartite Plan-Based Model of Dialogue*, PhD thesis, University of Delaware, 1993, Reproduced as TR 93-19 Department of Computer and Information Science, University of Delaware.

[Levelt, 1983] Willem J. M. Levelt, "Monitoring and self-repair in speech," *Cognition*, 14:41–104, 1983.

[Levelt and Cutler, 1983] Willem J. M. Levelt and A. Cutler, "Prosodic Markings in Speech Repair," *Journal of Semantics*, 2:205–217, 1983.

[Levesque *et al.*, 1990] Hector J. Levesque, Phillip R. Cohen, and Jose H. T. Nunes, "On Acting Together," In *Proceedings AAAI-90*, pages 94–99, 1990.

[Levinson, 1983] Stephen C. Levinson, *Pragmatics*, Cambridge University Press, 1983.

[Lewis, 1969] David K. Lewis, *Convention: A Philosophical Study*, Harvard University Press, 1969.

[Litman, 1985] Diane J. Litman, *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*, PhD thesis, University of Rochester, 1985, Reproduced as TR 170 Department of Computer Science, University of Rochester.

[Litman and Allen, 1990] Diane J. Litman and James F. Allen, "Discourse Processing and Common Sense Plans," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Luff *et al.*, 1990] Paul Luff, Nigel Gilbert, and David Frohlich, editors, *Computers and Conversation*, Academic Press, 1990.

[Maida, 1984] Anthony S. Maida, "Belief Spaces: Foundations of a computational Theory of Belief," Technical Report Cs-84-22, Pennsylvania State University Department of Computer Science, December 1984.

[Mann and Thompson, 1987] William C. Mann and Sandra A. Thompson, "Rhetorical Structure Theory: A Theory of Text Organization," Technical Report ISI/RS-87-190, USC, Information Sciences Institute, June 1987.

[McCarty, 1986] L. Thorne McCarty, "Permissions and Obligations: An Informal Introduction," Technical Report LRP-TR-19, Dept. of Computer Science, Rutgers University, 1986.

[McCawley, 1988] James D. McCawley, *The Syntactic Phenomena of English*, University of Chicago Press, 1988.

[McCawley, 1989] James D. McCawley, "Individuation in and of Syntactic Structures," In Mark Baltin and Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*, chapter 7, pages 117–138. University of Chicago Press, 1989.

[McLemore, 1991] Cynthia McLemore, *The Pragmatic Interpretation of English Intonation*, PhD thesis, University of Texas at Austin, 1991.

[McRoy, 1993] Susan McRoy, *Abductive Interpretation and Reinterpretation of Natural Language Utterances*, PhD thesis, University of Toronto, 1993, Reproduced as TR CSRI-288 Department of Computer Science, University of Toronto.

[Novick, 1988] David Novick, *Control of Mixed-Initiative Discourse Through Meta-Locutionary Acts: A Computational Model*, PhD thesis, University of Oregon, 1988, also available as U. Oregon Computer and Information Science Tech Report CIS-TR-88-18.

[Orestrom, 1983] Bengt Orestrom, *Turn-Taking in English Conversation*, Lund Studies in English: Number 66. CWK Gleerup, 1983.

[Perner and Garnham, 1988] Josef Perner and Alan Garnham, "Conditions for Mutuality," *Journal of Semantics*, 6:369–385, 1988.

[Perrault, 1984] C. Raymond Perrault, "On the Mathematical Properties of Linguistic Theories," *Computational Linguistics*, 10:165–176, 1984, Also appears in [?].

[Perrault, 1990] C. Raymond Perrault, "An Application of Default Logic to Speech Act Theory," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Perrault and Allen, 1980] C. Raymond Perrault and James F. Allen, "A Plan-Based Analysis of Indirect Speech Acts," *American Journal of Computational Linguistics*, 6(3-4):167–82, July-December 1980.

[Pierrehumbert, 1980] J. B. Pierrehumbert, "The Phonology and Phonetics of English Intonation," Doctoral dissertation, Massachusetts Institute of Technology, 1980.

[Pierrehumbert and Hirschberg, 1990] Janet Pierrehumbert and Julia Hirschberg, "The Meaning of Intonational Contours in the Interpretation of Discourse," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Poesio, 1991] Massimo Poesio, "Expectation-based Recognition of Intentional Structure," In *Working Notes AAAI Fall Symposium on Discourse Structure in Natural Language Understanding and Generation*, November 1991.

[Pollack, 1990] Martha E. Pollack, "Plans as Complex Mental Attitudes," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[Pollack, 1986] Martha E. Pollack, *Inferring Domain Plans in Question-Answering*, PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 1986.

[Rambow, 1993] Owen Rambow, editor, *Proceedings ACL SIG Workshop on Intentionality and Structure in Discourse Relations*, Association for Computational Linguistics, June 1993.

[Raudaskoski, 1990] Pirkko Raudaskoski, "Repair Work in Human-Computer Interaction," In Paul Luff, Nigel Gilbert, and David Frohlich, editors, *Computers and Conversation.* Academic Press, 1990.

[Reiter, 1990] Ehud Reiter, "The Computational Complexity of Avoiding Conversational Implicatures," In *Proceedings ACL-90*, pages 97–104, 1990.

[Reiter, 1980] R. Reiter, "A Logic for Default Reasoning," *AI*, 13(1,2):81–132, April 1980.

[Sacks *et al.*, 1974] H. Sacks, E. A. Schegloff, and G. Jefferson, "A Simplest Systematics For the organization of Turn-Taking for Conversation," *Language*, 50:696–735, 1974.

[Schegloff, 1987] Emmanuel A. Schegloff, "Recycled Turn-beginnings: A precise repair mechanism in conversation's turn-taking organization," In Graham Button and John R. E. Lee, editors, *Talk and Social Organization.* Multilingual Matters, 1987.

[Schegloff *et al.*, 1977] Emmanuel A. Schegloff, G. Jefferson, and H. Sacks, "The Preference for Self Correction in the Organization of Repair in Conversation," *Language*, 53:361–382, 1977.

[Schegloff and Sacks, 1973] Emmanuel A. Schegloff and H. Sacks, "Opening Up Closings," *Semiotica*, 7:289–327, 1973.

[Schiffer, 1972] Stephen R. Schiffer, *Meaning*, Oxford University Press, 1972.

[Searle, 1969] John R. Searle, *Speech Acts*, Cambridge University Press, New York, 1969.

[Searle, 1976] John R. Searle, "A classification of illocutionary acts," *Language in Society*, 5:1–23, 1976.

[Searle, 1990] John R. Searle, "Collective Intentions and Actions," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[Searle and Vanderveken, 1985] John R. Searle and Daniel Vanderveken, *Foundations of Illocutionary Logic*, Cambridge University Press, 1985.

[Sinclair and Coulthard, 1975] J. M. Sinclair and R. M. Coulthard, *Towards an analysis of Discourse: The English used by teachers and pupils.*, Oxford University Press, 1975.

[Stenstrom, 1984] Anna-Brita Stenstrom, *Questions and Responses*, Lund Studies in English: Number 68. Lund : CWK Gleerup, 1984.

[Suchman, 1987] Lucy A. Suchman, *Plans and Situated Actions*, Cambridge University Press, 1987.

[Traum, 1991] David R. Traum, "The Discourse Reasoner in TRAINS-90," TRAINS Technical Note 91-5, Computer Science Dept. University of Rochester, 1991.

[Traum and Allen, 1991] David R. Traum and James F. Allen, "Causative Forces in Multi-Agent Planning," In Y. Demazeau and J. P. Muller, editors, *Decentralized A.I. 2*, pages 89–105. Elsevier Science Publishers B. V., 1991.

[Traum and Hinkelman, 1992] David R. Traum and Elizabeth A. Hinkelman, "Conversation Acts in Task-oriented Spoken Dialogue," *Computational Intelligence*, 8(3):575–599, 1992, Special Issue on Non-literal language.

[von Wright, 1951] G. H. von Wright, "Deontic Logic," *Mind*, 60:1–15, 1951.

[Walker, 1993] Marilyn A. Walker, *Informational Redundancy and Resource Bounds in Dialogue*, PhD thesis, University of Pennsylvania, 1993.

[Walker and Whittaker, 1990] Marilyn A. Walker and Steve Whittaker, "Mixed Initiative in Dialogue: An Investigation into Discourse Segmentation," In *Proceedings ACL-90*, pages 70–78, 1990.

[Werner and Demazeau, 1992] E. Werner and Y. Demazeau, editors, *Decentralized A.I. 3*, Elsevier Science Publishers B. V., 1992.

[Yngve, 1970] Victor H. Yngve, "On Getting A Word In Edgewise," In *Papers from the Sixth Regional Meeting*, pages 567–78. Chicago Linguistic Society, 1970.

A Completion of Trace from Chapter 6

This section presents the rest of the annotated trace used as an example in Section 6.6. The first few utterances, as well as the complete dialogue and relevant contextual information are given in that section.

A.1 Interpreting utterance 5-1

USER: so we need an engine to move the boxcar. <kt>

As with utterance 3-3=6, this is interpreted as a sequence of two conversational events, with the core speech act hypotheses given in (66). The purpose clause of the utterance is interpreted as a supplementary suggestion, while the main clause is one of the informational acts and/or a suggestion.

(66)
```
(:SEQUENCE (:SURF-INTERP [CE719] [ST-ACCEPT-0019])
           (:SURF-INTERP [CE721]
                 (:AND (:OR (:EX-OR [ST-INFORM-0020] [ST-CHECK-0021] [ST-YNQ-0022])
                            [ST-SUGGEST-0023])
                       [ST-SUPP-SUG-0024])))
```

The content of [ST-ACCEPT-0019] is the same as that for [ST-ACCEPT-0006], given in (29). The content of the informational acts [ST-INFORM-0020], [ST-CHECK-0021], and [ST-YNQ-0022], is shown in (67)a, the content of [ST-SUGGEST-0023] is shown in (67)b, and the content of [ST-SUPP-SUG-0024] is shown in (67)c. The object [X244] is the discourse marker for the boxcar mentioned in Utterance 3-3=6.

(67) a.
```
(:NEED-REQUIRE [SYSHUM] (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE)))
```

b.
```
(:THE ?P*T-PLAN ?DM*T-ANYTHING (:CURRENT-PLAN ?P*T-PLAN )
      (:USES (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE))
             NIL
             ?P*T-PLAN))
```

c.
```
(:THE ?P*T-PLAN ?DM*T-PLAN (:CURRENT-PLAN ?P*T-PLAN)
      (:EVENT-IN
          (:LF-EXISTS ?L*T-MOVE [E713]
              (:APPLY
```

```
                    (:LAMBDA ?E*T-MOVE
                        (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                              (:ROLE ?E*T-MOVE :R-OBJECT [X244])))
                    ?L*T-MOVE)
                (:OCCURS ?L*T-MOVE))
            ?P*T-PLAN))
```

There are also two argumentation acts for this utterance. A :so act, similar to that shown in (34), as well as a :purpose act shown in (68), linking up the supplementary suggestion to the main acts of the utterance.

(68) `(:PURPOSE [ST-SUPP-SUG-0024] [CE721])`

[ST-ACCEPT-0019] is rejected, just as [ST-ACCEPT-0006] was, because there are no matching unaccepted acts performed by the system. In order to test [ST-INFORM-0020], the :PURPOSE relation becomes important, changing the query from that in (67)a to that given in (69).

(69)
```
(:NEED-FOR-PURPOSE
    (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE))
    (:LF-EXISTS ?L*T-MOVE [E713]
        (:APPLY
            (:LAMBDA ?E*T-MOVE
                (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                      (:ROLE ?E*T-MOVE :R-OBJECT [X244])))
            ?L*T-MOVE)
        (:OCCURS ?L*T-MOVE)))
```

The proof is successful, since, in fact, an engine *is* needed to move the mentioned boxcar (or any boxcar), since there are no other methods for moving boxcars in the TRAINS domain that don't involve an engine role. This rules out the inform possibility, and the remaining informational act is [ST-CHECK-0021].

Evaluating [ST-SUGGEST-0023] leads to another *incorporate* call to the domain plan reasoner. This will also be of :incorp-type :role-filler, with a focus on the non-specific engine predicate. The :purpose argumentation act will add a :purpose argument, which the plan reasoner will first incorporate, before incorporating the main act within its scope. The :so argumentation act will also add a :context argument, however, in this case there are multiple possibilities for the relevant context. First, the system tries the most recent planner return, that coming from utterance 3-7. Since [ST-SUGGEST-0015] involved filling in the ORANGES role in the move-commodity event, this event is given as the context. The first call is given in (70).

(70)
```
(INCORPORATE
    (:USE-OBJECT (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE)))
    [PLAN-4719]
    :INCORP-TYPE :ROLE-FILLER
    :CONTEXT #EVENT-GOAL<[MOVE-COMMODITY-4779]>
    :PURPOSE (:LF-EXISTS ?L*T-MOVE [E713]
                (:APPLY
```

```
                    (:LAMBDA ?E*T-MOVE
                         (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                               (:ROLE ?E*T-MOVE :R-OBJECT [X244])))
                    ?L*T-MOVE)
                (:OCCURS ?L*T-MOVE))
    :FOCUS (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE)))
```

In incorporating the purpose move-car event, an ambiguity is noticed: this move-car event could either be the one which generates the move-commodity event, or the move-car event which moves the boxcar to Corning. While this ambiguity would not be fatal, before engaging heuristics to choose between these events, the system tries another call, using utterance 3-3=6, instead for the intended previous event of the :so act. This yields the incorporate call in (71), which succeeds unambiguously, adding the propositions in (72).

(71)
```
(INCORPORATE
   (:USE-OBJECT (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE)))
   [PLAN-4719]
   :INCORP-TYPE :ROLE-FILLER
   :CONTEXT #EVENT<[MOVE-CAR-7867]>
   :PURPOSE (:LF-EXISTS ?L*T-MOVE [E713]
                (:APPLY
                   (:LAMBDA ?E*T-MOVE
                        (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                              (:ROLE ?E*T-MOVE :R-OBJECT [X244])))
                   ?L*T-MOVE)
                (:OCCURS ?L*T-MOVE))
   :FOCUS (:LAMBDA ?o*T-ENGINE (:DISC-MARKER [X710] ?o*T-ENGINE)))
```

(72)
```
(:PLAN-FACT (:AND (:DISC-MARKER [X710] [F-ENGINE MOVE-CAR-7867])) [PLAN-4719])
(:PLAN-SUPPORTS (:AND (:DISC-MARKER [X710] [F-ENGINE MOVE-CAR-7867]))
      [MOVE-CAR-7867] [PLAN-4719])
(:PLAN-PREMISE (:AND (:DISC-MARKER [X710] [F-ENGINE MOVE-CAR-7867])) [PLAN-4719])
(:AND (:DISC-MARKER [X710] [F-ENGINE MOVE-CAR-7867]))
```

This call also validates the supplementary suggestion which is the *purpose* of the suggestion. The validated core speech acts for this utterance are given in (73)a, while the validated argumentation acts are shown in (73)b

(73) a. `[ST-CHECK-0021] [ST-SUGGEST-0023] [ST-SUPP-SUG-0024]`

b. `(:SO [CE251] [CE721]) (:PURPOSE [ST-SUPP-SUG-0024] [CE721])`

This utterance will be seen as an :init grounding act, adding a new DU (DU-3) to the stack. The conversation acts for this utterance are shown in (74) with the resulting discourse state shown in Figure A.1.

(74)
```
:CSAS [ST-CHECK-0021] [ST-SUGGEST-0023] [ST-SUPP-SUG-0024]
:ARGS (:SO [CE251] [CE721]) (:PURPOSE [ST-SUPP-SUG-0024] [CE721])
:GAS (:INIT MANAGER DU-3 ([ST-CHECK-0021] [ST-SUGGEST-0023] [ST-SUPP-SUG-0024]) NIL)
:TTAS (KEEP-TURN MANAGER [NOW9]) (KEEP-TURN MANAGER [NOW9])
```

```
Turn Holder: Manager
DU Stack:
 DU-3  :INITIATOR MANAGER :STATE 1      :CONTEXT #<context SB-G11668>
       :CSA-LIST ([ST-CHECK-0021] [ST-SUGGEST-0023] [ST-SUPP-SUG-0024])
       :PROCESSED NIL
 DU-2  :INITIATOR MANAGER :STATE F      :CONTEXT #<context SB-G8281>
       :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
 DU-1  :INITIATOR MANAGER :STATE F      :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
```

Figure A.1: Discourse Context after utterance 5-1

A.2 Interpreting utterance 5-2

USER: right?

Analysis of this utterance is exactly parallel to that for utterance 3-8, given in Section 6.6.7. This is seen as a `:repair` grounding act, replacing `[ST-CHECK-0021]` with `[ST-CHECK-0025]` in `DU-3`. The set of conversation acts is shown in (75).

(75)
```
:CSAS [ST-CHECK-0025] :ARGS NIL
:GAS (:REPAIR MANAGER DU-3 ([ST-CHECK-0025]) ([ST-CHECK-0021]))
:TTAS (KEEP-TURN MANAGER [NOW10]) (RELEASE-TURN MANAGER [NOW10])
```

A.3 Acting after utterance 5-2, and Utterance 6

The system will act for the most part as described in Section 6.6.8. First the system will decide to acknowledge the acts in `DU-3`, then it will decide to meet the obligation to check-if, by responding with utterance 6. The Discourse context just prior to utterance 6 is shown in Figure A.2.

SYSTEM: Right.

This is interpreted the same as utterance 4 – as an `:ack` of the top DU, as well as accepting the proposals in `[ST-SUGGEST-0023]` and `[ST-SUPP-SUG-0024]`, and releasing the turn. The conversation acts are shown in (76) and the new discourse state is shown in Figure A.3.

(76)
```
:CSAS [ST-ACCEPT-0026] :ARGS NIL
:GAS (:ACK SYSTEM DU-3 ([ST-ACCEPT-0026]) NIL)
:TTAS ((KEEP-TURN SYSTEM [NOW11]) (RELEASE-TURN SYSTEM [NOW11])
```

A.4 Interpreting utterances 7-1=2, 7-3, and 8

USER: so there's an engine at Avon. <kt>

```
Discourse Obligations: (:CHECK-IF (:RIGHT-CORRECT [ST-CHECK-0021]))
Turn Holder: System
Intended Speech Acts:
  (:ACK ([ST-CHECK-0025] [ST-SUGGEST-0023] [ST-SUPP-SUG-0024]) GROUNDING)
  (:INFORM-IF (:FOCUS :RIGHT (:RIGHT-CORRECT [ST-CHECK-0021]))
              (:OBLIGATION (:CHECK-IF (:RIGHT-CORRECT [ST-CHECK-0021]))))
DU Stack:
 DU-3  :INITIATOR MANAGER :STATE 1       :CONTEXT #<context SB-G11668>
       :CSA-LIST ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025])
       :PROCESSED ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025])
 DU-2  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G8281>
       :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
 DU-1  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
Unaccepted Proposals: [ST-SUGGEST-0023] [ST-SUPP-SUG-0024]
```

Figure A.2: Discourse Context when uttering 6

```
Turn Holder: Manager
DU Stack:
 DU-3  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G11668>
       :CSA-LIST ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025] [ST-ACCEPT-0026])
       :PROCESSED ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025] [ST-ACCEPT-0026])
 DU-2  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G8281>
       :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                  [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
 DU-1  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G4940>
       :CSA-LIST ([ST-INFORM-0001] [ST-SUGGEST-0004] [ST-ACCEPT-0005])
```

Figure A.3: Discourse Context after utterance 6

Interpretation of this utterance proceeds in the manner described above for previous utterances. The surface core speech act hypotheses are given in (77)a. The content of the informational acts is given in (77)b, while the content of [ST-SUGGEST-0032] is given in (77)c. There is also a :so argumentation act similar to that in (34).

```
(77) a. (:SEQUENCE (:SURF-INTERP [CE993] [ST-ACCEPT-0028])
                   (:SURF-INTERP [CE995]
                        (:OR (:EX-OR [ST-INFORM-0029] [ST-CHECK-0030] [ST-YNQ-0031])
                             [ST-SUGGEST-0032])))

     b. (:LF-EXISTS ?v13237*T-ENGINE [X989] NIL (:AT ?v13237*T-ENGINE [AVON] [E991]))

     c. (:THE ?P*T-PLAN ?DM*T-ANYTHING (:CURRENT-PLAN ?P*T-PLAN)
              (:LF-EXISTS ?VAR->?foo*T-ENGINE [X989] NIL
                   (:USES ?VAR*T-ENGINE (:AT ?VAR*T-ENGINE [AVON] [E991]) ?P*T-PLAN)))
```

As with previous analyses, [ST-ACCEPT-0028] is ruled out because there is nothing to accept, [ST-INFORM-0029] is ruled out because the system did find an engine at Avon in the **Shared** belief context ([ENGINE-1]).

Evaluating [ST-SUGGEST-0032] leads to the *incorporate* call in (78)a, which returns the results in (78)b, stating that this engine will fill the ENGINE role of the move-car event.

```
(78) a. (INCORPORATE (:USE-OBJECT [X989]) [PLAN-4719] :INCORP-TYPE :ROLE-FILLER
        :CONTEXT #EVENT-OLD<[MOVE-CAR-7867]> :FOCUS [X989]
        :BG (:AT [X989] [AVON] [E991]))

     b. (:PLAN-FACT (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867])) [PLAN-4719])
        (:PLAN-SUPPORTS (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867]))
              [MOVE-CAR-7867] [PLAN-4719])
        (:PLAN-PREMISE (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867])) [PLAN-4719])
        (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867]))
```

The conversation acts recognized are shown in (79), and the updated discourse context is shown in Figure A.4. Note that DU-1 has dropped from the bottom of the DU stack. The core speech acts from this utterance may no longer be influenced by successive grounding acts. Any change in this material must be performed by reintroducing the material in new DUs.

```
(79) :CSAS [ST-CHECK-0030] [ST-SUGGEST-0032]
     :ARGS (:SO [CE721] [CE995])
     :GAS (:INIT MANAGER DU-4 ([ST-CHECK-0030] [ST-SUGGEST-0032]) NIL)
     :TTAS (KEEP-TURN MANAGER [NOW13]) (KEEP-TURN MANAGER [NOW13])
```

Turn Holder: Manager
DU Stack:

```
DU-4  :INITIATOR MANAGER :STATE 1      :CONTEXT #<context SB-G13784>
      :CSA-LIST ([ST-CHECK-0030] [ST-SUGGEST-0032])    :PROCESSED NIL
DU-3  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G11668>
      :CSA-LIST ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025] [ST-ACCEPT-0026])
DU-2  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G8281>
      :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                 [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
```

Unaccepted Proposals: [ST-SUGGEST-0023] [ST-SUPP-SUG-0024]

Figure A.4: Discourse Context after utterance 7-1=2

USER: right?

Analysis of utterance 7-3 proceeds in the same manner as utterances 3-8 (Section 6.6.7) and 5-2. The conversation acts are shown in (80).

```
(80) :CSAS [ST-CHECK-0033] :ARGS NIL
     :GAS (:REPAIR MANAGER DU-4 ([ST-CHECK-0033]) ([ST-CHECK-0030]))
     :TTAS (KEEP-TURN MANAGER [NOW14]) (RELEASE-TURN MANAGER [NOW14])
```

The actions afterward will also parallel those after those utterances, (as described in Section 6.6.8). This will lead to utterance 8:

SYSTEM: Right.

Which will be interpreted just as utterances 4 (Section 6.6.9) and 6, with the set of conversation acts shown in (81). The discourse context after interpreting utterance 8 is shown in Figure A.5.

(81)
```
:CSAS [ST-ACCEPT-0034] :ARGS NIL
:GAS (:ACK SYSTEM DU-4 ([ST-ACCEPT-0034]) NIL)
:TTAS (KEEP-TURN SYSTEM [NOW15]) (RELEASE-TURN SYSTEM [NOW15])
```

Turn Holder: Manager
DU Stack:

```
DU-4  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G13784>
      :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
      :PROCESSED ([ST-CHECK-0033] [ST-SUGGEST-0032] [ST-ACCEPT-0034])
DU-3  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G11668>
      :CSA-LIST ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025])
DU-2  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G8281>
      :CSA-LIST ([ST-INFORM-0007] [ST-SUGGEST-0010] [ST-SUPP-INF-0011]
                 [ST-SUGGEST-0015] [ST-CHECK-0016] [ST-ACCEPT-0017])
```

Figure A.5: Discourse Context after utterance 8

A.5 Interpreting utterance 9=13

USER: so we should move the engine at Avon, engine E1, to Dansville to pick up the boxcar there.

The core speech act hypotheses for this utterance are shown in (82). Also, there are `:so` and `:purpose` argumentation acts similar to Utterance 5-1.

(82)
```
(:SEQUENCE (:SURF-INTERP [CE1206] [ST-ACCEPT-0036])
           (:SURF-INTERP [CE1208]
                (:AND (:OR (:EX-OR [ST-INFORM-0037] [ST-CHECK-0038] [ST-YNQ-0039])
                           [ST-SUGGEST-0040])
                      [ST-SUPP-SUG-0041])))
```

The content of the informational acts is given in (83). The "should" translates literally to a *weak obligation* – something that *should* be done, as opposed to an *obligation* which is something that *must* be done.

(83)
```
(:WEAK-OBLIG [SYSHUM]
        (:LF-EXISTS ?L*T-MOVE [E1193]
            (:APPLY (:LAMBDA ?E*T-MOVE
                        (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                              (:ROLE ?E*T-MOVE :R-OBJECT [ENGINE-1])
                              (:ROLE ?E*T-MOVE :R-DST [DANSVILLE])))
                    ?L*T-MOVE)
            (:OCCURS ?L*T-MOVE))
        [E1204])
```

The content of [ST-SUGGEST-0040] is given in (84). That of [ST-SUPP-SUG-0041] is given in (85).

```
(84) (:THE ?P*T-PLAN ?DM*T-PLAN (:CURRENT-PLAN ?P*T-PLAN)
          (:EVENT-IN
              (:LF-EXISTS ?L*T-MOVE [E1193]
                  (:APPLY (:LAMBDA ?E*T-MOVE
                                (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                                      (:ROLE ?E*T-MOVE :R-OBJECT [ENGINE-1])
                                      (:ROLE ?E*T-MOVE :R-DST [DANSVILLE])))
                          ?L*T-MOVE)
                  (:OCCURS ?L*T-MOVE))
              ?P*T-PLAN))
```

```
(85) (:THE ?P*T-PLAN ?DM*T-PLAN (:CURRENT-PLAN ?P*T-PLAN)
          (:EVENT-IN
              (:LF-EXISTS ?L*T-COUPLE [E1199]
                  (:APPLY (:LAMBDA ?E*T-COUPLE
                                (:AND (:ROLE ?E*T-COUPLE :R-AGENT [SYSHUM])
                                      (:ROLE ?E*T-COUPLE :R-CAR [BOXCAR-1])))
                          ?L*T-COUPLE)
                  (:OCCURS ?L*T-COUPLE))
              ?P*T-PLAN))
```

[ST-ACCEPT-0036] is rejected because there is nothing to accept. [ST-INFORM-0037] is deemed applicable, since the system is not aware of any such weak obligation. Evaluation of [ST-SUGGEST-0040] results in the call to the domain plan reasoner shown in (86), which returns the additions to the plan given in (87).

```
(86) (INCORPORATE
        (:LF-EXISTS ?L*T-MOVE [E1193]
            (:APPLY (:LAMBDA ?E*T-MOVE
                          (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYSHUM])
                                (:ROLE ?E*T-MOVE :R-OBJECT [ENGINE-1])
                                (:ROLE ?E*T-MOVE :R-DST [DANSVILLE])))
                    ?L*T-MOVE)
            (:OCCURS ?L*T-MOVE))
        [PLAN-4719]
        :INCORP-TYPE :EVENT :CONTEXT #EVENT-OLD<[MOVE-CAR-7867]>
        :PURPOSE
        (:LF-EXISTS ?L*T-COUPLE [E1199]
            (:APPLY (:LAMBDA ?E*T-COUPLE
                          (:AND (:ROLE ?E*T-COUPLE :R-AGENT [SYSHUM])
                                (:ROLE ?E*T-COUPLE :R-CAR [BOXCAR-1])))
                    ?L*T-COUPLE)
            (:OCCURS ?L*T-COUPLE)))
```

```
(87) (:PLAN-FACT    (:COUPLED [F-ENGINE MOVE-CAR-7867] [F-CAR MOVE-CAR-7867]
                              [F-SRC MOVE-CAR-7867] [F-PRE2 MOVE-CAR-7867]) [PLAN-4719])
     (:PLAN-ENABLES (:COUPLED [F-ENGINE MOVE-CAR-7867] [F-CAR MOVE-CAR-7867]
                              [F-SRC MOVE-CAR-7867] [F-PRE2 MOVE-CAR-7867])
                    [MOVE-CAR-7867] [PLAN-4719])
```

```
(:PLAN-EVENT [COUPLE-16190] [PLAN-4719])
(:PLAN-ACHIEVES [COUPLE-16190]
                (:COUPLED [F-ENGINE MOVE-CAR-7867] [F-CAR MOVE-CAR-7867]
                          [F-SRC MOVE-CAR-7867] [F-PRE2 MOVE-CAR-7867]) [PLAN-4719])
(:PLAN-FACT     (:AT [F-ENGINE COUPLE-16190] [F-LOCATION COUPLE-16190]
                          [F-PRE1 COUPLE-16190]) [PLAN-4719])
(:PLAN-ENABLES (:AT [F-ENGINE COUPLE-16190] [F-LOCATION COUPLE-16190]
                          [F-PRE1 COUPLE-16190])
                [COUPLE-16190] [PLAN-4719])
(:PLAN-EVENT [MOVE-ENGINE-16503] [PLAN-4719])
(:PLAN-ACHIEVES[MOVE-ENGINE-16503]
                (:AT [F-ENGINE COUPLE-16190] [F-LOCATION COUPLE-16190]
                          [F-PRE1 COUPLE-16190]) [PLAN-4719])
(:COUPLED       [F-ENGINE MOVE-CAR-7867] [F-CAR MOVE-CAR-7867]
                [F-SRC MOVE-CAR-7867] [F-PRE2 MOVE-CAR-7867])
(:AT            [F-ENGINE COUPLE-16190] [F-LOCATION COUPLE-16190]
                [F-PRE1 COUPLE-16190])
```

This also results in validating [ST-SUPP-SUG-0041], as well as the :so and :purpose argumentation acts. The final set of conversation acts is shown in (88).

(88)
```
:CSAS [ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041]
:ARGS (:SO [CE995] [CE1208]) (PURPOSE [ST-SUPP-SUG-0041][CE1208])
:GAS (INIT MANAGER DU-5 ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041]) NIL)
:TTAS (KEEP-TURN MANAGER [NOW17]) (RELEASE-TURN MANAGER [NOW17])
```

A.6 Producing and interpreting utterance 14

The reasoning here is similar to that after Utterance 1 (Section 6.6.3). First the system will decide to acknowledge the acts in the top DU, and then to accept them, leading to production of utterance 14. The discourse state just before the production of utterance 14 is shown in Figure A.6.

Turn Holder: Manager
Intended Speech Acts:
(:ACK ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041]) GROUNDING)
(:ACCEPT [ST-INFORM-0037]) (:ACCEPT [ST-SUGGEST-0040]) (:ACCEPT [ST-SUPP-SUG-0041])
DU Stack:

```
DU-5  :INITIATOR MANAGER :STATE 1       :CONTEXT #<context SB-G17094>
      :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041])
      :PROCESSED ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041])
DU-4  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G13784>
      :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
DU-3  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G11668>
      :CSA-LIST ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025])
```

Unaccepted Proposals: [ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041]

Figure A.6: Discourse Context before utterance 14

SYSTEM: Okay.

Utterance 14 is interpreted just as utterance 2, as described in Section 6.6.4. This utterance is seen as an acceptance of [ST-INFORM-0037], [ST-SUGGEST-0040], and [ST-SUPP-SUG-0041]. The conversation acts for this utterance are shown in (89), and the resulting discourse context is shown in Figure A.7.

(89)
```
:CSAS ([ST-ACCEPT-0042]) :ARGS NIL
:GAS (:ACK SYSTEM DU-5 ([ST-ACCEPT-0042]) NIL))
:TTAS (KEEP-TURN SYSTEM [NOW18]) (RELEASE-TURN SYSTEM [NOW18])
```

Turn Holder: Manager
DU Stack:

```
DU-5  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G17094>
      :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
      :PROCESSED ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
DU-4  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G13784>
      :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
DU-3  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G11668>
      :CSA-LIST ([ST-SUGGEST-0023] [ST-SUPP-SUG-0024] [ST-CHECK-0025])
```

Figure A.7: Discourse Context after utterance 14

A.7 Interpreting Utterance 15-2=4

USER: and move it from Dansville to Corning. <kt>

The "and" is treated as a discourse marker, meriting its own utterance event, just as for "so", above. Like those utterances, this will have a potential *accept* interpretation at the core speech act level. The "and" also yields a :and argumentation act. The imperative form of the main clause yields a *request* interpretation. The core speech act hypotheses are shown in (90)a. (90)b shows the content of [ST-REQUEST-0044]. The content of [ST-ACCEPT-0043] is the same as (29). The :and act is shown in (90)c.

(90) a.
```
(:SEQUENCE (:SURF-INTERP [CE1646] [ST-ACCEPT-0043])
           (:SURF-INTERP [CE1648] [ST-REQUEST-0044]))
```

b.
```
(:THE ?P*T-PLAN ?DM*T-PLAN (:CURRENT-PLAN ?P*T-PLAN)
      (:EVENT-IN (:LF-EXISTS ?L*T-MOVE [E1649]
                             (:APPLY
                                 (:LAMBDA ?E*T-MOVE
                                     (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYS])
                                           (:ROLE ?E*T-MOVE :R-OBJECT [BOXCAR-1])
                                           (:ROLE ?E*T-MOVE :R-DST [CORNING])
                                           (:ROLE ?E*T-MOVE :R-SRC [DANSVILLE])))
                                 ?L*T-MOVE)
                             (:OCCURS ?L*T-MOVE))
                 ?P*T-PLAN))
```

c.
```
(:THE ?PA*T-SPEECHACT ?PADM*T-ANYTHING (:PREV-CE [CE1646] ?PA*T-SPEECHACT)
      (:THE ?NA*T-SPEECHACT ?NADM*T-ANYTHING (:NEXT-CE [CE1646] ?NA*T-SPEECHACT)
            (:AND ?PA*T-SPEECHACT ?NA*T-SPEECHACT)))
```

[ST-ACCEPT-0043] is ruled out because there is nothing to accept, while [ST-REQUEST-0044] leads to an *incorporate* call to the domain plan reasoner. The :and act is seen as a *temporal* connective, making the next event referred to follow the previous event. As with the :so acts, the pruner first tries the node from the previous suggestion, 9=13. The planner call is shown in (91).

(91)
```
(INCORPORATE
    (:LF-EXISTS ?L*T-MOVE [E1649]
        (:APPLY
            (:LAMBDA ?E*T-MOVE
                (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYS])
                      (:ROLE ?E*T-MOVE :R-OBJECT [BOXCAR-1])
                      (:ROLE ?E*T-MOVE :R-DST [CORNING])
                      (:ROLE ?E*T-MOVE :R-SRC [DANSVILLE])))
            ?L*T-MOVE)
        (:OCCURS ?L*T-MOVE))
    [PLAN-4719]
    :INCORP-TYPE :EVENT
    :TEMP-SEQ #EVENT<[MOVE-ENGINE-16503]>)
```

The incorporate call is successful, matching this event with a move-car event already in the plan, [MOVE-CAR-7867]. This call ends up specifying further the agent role from [SYSHUM] to [SYS][1], adding a *Source* of [DANSVILLE], and specifying that this will take place after the move-engine event mentioned in utterance 9=13. The new additions are shown in (92)

(92)
```
(:PLAN-FACT (:AND (:ROLE [MOVE-CAR-7867] :R-AGENT [SYS])
                  (:ROLE [MOVE-CAR-7867] :R-OBJECT [BOXCAR-1])
                  (:ROLE [MOVE-CAR-7867] :R-DST [CORNING])
                  (:ROLE [MOVE-CAR-7867] :R-SRC [DANSVILLE])
                  (:TIME-RELN [F-TIME [MOVE-ENGINE-16503]]
                              (:B :M) [F-TIME [MOVE-CAR-7867]]))
            [PLAN-4719])
```

This utterance is the initiation of a new DU, DU-6. The conversation acts for this utterance are shown in (93), and the resulting discourse context is shown in Figure A.8.

(93)
```
:CSAS [ST-REQUEST-0044] :ARGS (:AND [CE1208] [CE1648])
:GAS (INIT MANAGER DU-6 ([ST-REQUEST-0044]) NIL)
:TTAS (KEEP-TURN MANAGER [NOW20]) (KEEP-TURN MANAGER [NOW20])
```

A.8 Interpreting Utterance 15-5=7

USER: load up some oranges into the boxcar. <kt>

The only interpretation for this utterance is a request, shown in (94)a, with content shown in (94)b.

[1] Actually, this is still a case of indirect agency, and the engine will be the direct agent.

```
Turn Holder: Manager
DU Stack:
 DU-6  :INITIATOR MANAGER :STATE 1       :CONTEXT #<context SB-G18980>
       :CSA-LIST ([ST-REQUEST-0044]) :PROCESSED NIL
 DU-5  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G17094>
       :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
 DU-4  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G13784>
       :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
```

Figure A.8: Discourse Context after utterance 15-2=4

```
(94) a. (:SURF-INTERP [CE1790] [ST-REQUEST-0045])

     b. (:THE ?P*T-PLAN ?DM*T-PLAN (:CURRENT-PLAN ?P*T-PLAN)
           (:EVENT-IN
               (:LF-EXISTS ?v19300*T-ORANGES [X1785] NIL
                   (:LF-EXISTS ?L*T-LOAD [E1791]
                       (:APPLY
                           (:LAMBDA ?E*T-LOAD
                                (:AND (:ROLE ?E*T-LOAD :R-AGENT [SYS])
                                      (:ROLE ?E*T-LOAD :R-COMMODITY ?v19300*T-ORANGES)
                                      (:ROLE ?E*T-LOAD :R-CONTAINER [X714])))
                           ?L*T-LOAD)
                       (:OCCURS ?L*T-LOAD)))
               ?P*T-PLAN))
```

Evaluating [ST-REQUEST-0045] yields the domain planner call in (95), which succeeds, resulting in the additions in (96).

```
(95) (INCORPORATE
       (:LF-EXISTS ?v19300*T-ORANGES [X1785] NIL
           (:LF-EXISTS ?L*T-LOAD [E1791]
               (:APPLY
                   (:LAMBDA ?E*T-LOAD
                        (:AND (:ROLE ?E*T-LOAD :R-AGENT [SYS])
                              (:ROLE ?E*T-LOAD :R-COMMODITY ?v19300*T-ORANGES)
                              (:ROLE ?E*T-LOAD :R-CONTAINER [X714])))
                   ?L*T-LOAD)
               (:OCCURS ?L*T-LOAD)))
       [PLAN-4719]
       :INCORP-TYPE :EVENT)

(96) (:PLAN-FACT     (:IN  [F-COMMODITY MOVE-COMMODITY-4779] [F-CAR MOVE-COMMODITY-4779]
                           [F-PRE2 MOVE-COMMODITY-4779])
                     [PLAN-4719])
     (:PLAN-ENABLES  (:IN  [F-COMMODITY MOVE-COMMODITY-4779] [F-CAR MOVE-COMMODITY-4779]
                           [F-PRE2 MOVE-COMMODITY-4779])
                     [MOVE-COMMODITY-4779] [PLAN-4719])
     (:PLAN-EVENT    [LOAD-ORANGES-21221] [PLAN-4719])
     (:PLAN-ACHIEVES [LOAD-ORANGES-21221]
                     (:IN [F-COMMODITY MOVE-COMMODITY-4779] [F-CAR MOVE-COMMODITY-4779]
                           [F-PRE2 MOVE-COMMODITY-4779])
```

```
                [PLAN-4719])
(:PLAN-FACT     (:AND (:ROLE [LOAD-ORANGES-21221] :R-AGENT [SYS])
                      (:ROLE [LOAD-ORANGES-21221] :R-COMMODITY [X1785])
                      (:ROLE [LOAD-ORANGES-21221] :R-CONTAINER [X714]))
                [PLAN-4719])
(:IN            [F-COMMODITY MOVE-COMMODITY-4779] [F-CAR MOVE-COMMODITY-4779]
                [F-PRE2 MOVE-COMMODITY-4779])
```

In addition to adding the load-oranges event to the plan, these additions specify that this event achieves the fact of the oranges being in the car, which is a precondition of the move-commodity action.

This utterance is a *continuation* of DU-6, and the conversation acts for this utterance are shown in (97), resulting in the discourse state in Figure A.9.

(97)
```
:CSAS [ST-REQUEST-0045] :ARGS NIL
:GAS (:CONT MANAGER DU-6 ([ST-REQUEST-0045]) NIL)
:TTAS (KEEP-TURN MANAGER [NOW21]) (KEEP-TURN MANAGER [NOW21])
```

Turn Holder: Manager
DU Stack:

```
DU-6  :INITIATOR MANAGER :STATE 1       :CONTEXT #<context SB-G18980>
      :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045]) :PROCESSED NIL
DU-5  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G17094>
      :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
DU-4  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G13784>
      :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
```

Figure A.9: Discourse Context after utterance 15-5=7

A.9 Interpreting Utterance 15-8=10

USER: and then move it on to Bath.

This utterance is very similar to 15-2=4, yielding the core speech act hypotheses in (98)a, where the content of [ST-REQUEST-0047] is shown in (98)b. There is also an :and-then argumentation act, shown in (98)c, which is like the :and act, except unambiguously temporal sequencing.

(98) a.
```
(:SEQUENCE (:SURF-INTERP [CE1906] [ST-ACCEPT-0046])
           (:SURF-INTERP [CE1908] [ST-REQUEST-0047]))
```

b.
```
(:THE ?P*T-PLAN ?DM*T-PLAN (:CURRENT-PLAN ?P*T-PLAN)
      (:EVENT-IN (:LF-EXISTS ?L*T-MOVE [E1909]
                             (:APPLY
                                 (:LAMBDA ?E*T-MOVE
                                     (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYS])
                                           (:ROLE ?E*T-MOVE :R-OBJECT [BOXCAR-1])
                                           (:ROLE ?E*T-MOVE :R-DST [BATH])))
```

```
                                    ?L*T-MOVE)
                                (:OCCURS ?L*T-MOVE))
                            ?P*T-PLAN))

    c. (:THE ?PA*T-SPEECHACT ?PADM*T-ANYTHING (:PREV-CE [CE1646] ?PA*T-SPEECHACT)
            (:THE ?NA*T-SPEECHACT ?NADM*T-ANYTHING (:NEXT-CE [CE1646] ?NA*T-SPEECHACT)
                (:AND-THEN ?PA*T-SPEECHACT ?NA*T-SPEECHACT)))
```

As happened with previous hypotheses, [ST-ACCEPT-0046] is rejected, and [ST-REQUEST-0047] leads to the incorporate call shown in (99).

(99)
```
(INCORPORATE
   (:LF-EXISTS ?L*T-MOVE [E1909]
      (:APPLY
         (:LAMBDA ?E*T-MOVE
            (:AND (:ROLE ?E*T-MOVE :R-AGENT [SYS])
                  (:ROLE ?E*T-MOVE :R-OBJECT [BOXCAR-1])
                  (:ROLE ?E*T-MOVE :R-DST [BATH])))
         ?L*T-MOVE)
      (:OCCURS ?L*T-MOVE))
   [PLAN-4719]
   :INCORP-TYPE :EVENT
   :TEMP-SEQ #EVENT<[LOAD-ORANGES-21221]>)
```

This incorporation is successful – the mentioned event is the move-car event which generates the top move-commodity event. This incorporation specifies the agent role, as above, and also specifies the time as occurring after the load-oranges event.

(100)
```
(:PLAN-FACT (:AND (:ROLE [F-MOVE-CAR MOVE-COMMODITY-4779] :R-AGENT [SYS])
                  (:ROLE [F-MOVE-CAR MOVE-COMMODITY-4779] :R-OBJECT [BOXCAR-1])
                  (:ROLE [F-MOVE-CAR MOVE-COMMODITY-4779] :R-DST [BATH])
                  (:TIME-RELN [F-TIME [LOAD-ORANGES-21221]]
                              (:B :M) [F-TIME [F-MOVE-CAR MOVE-COMMODITY-4779]]))
            [PLAN-4719])
```

This utterance is also a continue of DU-6, but releases the turn to the system. The conversation acts for this utterance are shown in (101) and the resulting discourse context is shown in Figure A.10.

(101)
```
:CSAS [ST-REQUEST-0047] :ARGS (:AND-THEN [CE1790] [CE1908])
   :GAS (:CONT MANAGER DU-6 ([ST-REQUEST-0047]) NIL)
   :TTAS (KEEP-TURN MANAGER [NOW23]) (RELEASE-TURN MANAGER [NOW23])
```

A.10 Producing and interpreting utterance 16

First, the system decides to acknowledge the acts in DU-6, leading to the Discourse context shown in Figure A.11.

```
Turn Holder: System
DU Stack:
 DU-6  :INITIATOR MANAGER :STATE 1        :CONTEXT #<context SB-G18980>
       :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047])  :PROCESSED NIL
 DU-5  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G17094>
       :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
 DU-4  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G13784>
       :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
```

Figure A.10: Discourse Context after utterance 15-8=10

```
Discourse Obligations: (ADDRESS [ST-REQUEST-0044]) (ADDRESS [ST-REQUEST-0045])
                       (ADDRESS [ST-REQUEST-0047])
Turn Holder: System
Intended Speech Acts:
    (:ACK ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047]) GROUNDING)
DU Stack:
 DU-6  :INITIATOR MANAGER :STATE 1        :CONTEXT #<context SB-G18980>
       :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047])
       :PROCESSED ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047])
 DU-5  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G17094>
       :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
 DU-4  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G13784>
       :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
Unaccepted Proposals: [ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047]
```

Figure A.11: Discourse Context after deciding to acknowledge DU-6

```
Discourse Obligations: (ADDRESS [ST-REQUEST-0044]) (ADDRESS [ST-REQUEST-0045])
                       (ADDRESS [ST-REQUEST-0047])
Turn Holder: System
Intended Speech Acts:
    (:ACK ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047]) GROUNDING)
    (:ACCEPT [ST-REQUEST-0044] (OBLIGATION (ADDRESS [ST-REQUEST-0044])))
    (:ACCEPT [ST-REQUEST-0045] (OBLIGATION (ADDRESS [ST-REQUEST-0045])))
    (:ACCEPT [ST-REQUEST-0047] (OBLIGATION (ADDRESS [ST-REQUEST-004])))
DU Stack:
 DU-6  :INITIATOR MANAGER :STATE 1        :CONTEXT #<context SB-G18980>
       :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047])
       :PROCESSED ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047])
 DU-5  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G17094>
       :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
 DU-4  :INITIATOR MANAGER :STATE F        :CONTEXT #<context SB-G13784>
       :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
       :PROCESSED ([ST-CHECK-0033] [ST-SUGGEST-0032] [ST-ACCEPT-0034])
Unaccepted Proposals: [ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047]
```

Figure A.12: Discourse Context before producing utterance 16

Next, the system will act on the obligations, one at a time, for each, deciding to accept the request. After all of the obligations have been addressed, the discourse context will be as shown in Figure A.12, at which point the system will produce utterance 16.

SYSTEM: Okay.

This utterance is interpreted as an acceptance of the three requests, discharging the discourse obligations and adding their contents to the shared plan. The conversation acts for this utterance are shown in (102), and the resulting discourse context in Figure A.13.

(102)
```
:CSAS [ST-ACCEPT-0048] :ARGS NIL
:GAS (:ACK SYSTEM DU-6 ([ST-ACCEPT-0048]) NIL)
:TTAS (KEEP-TURN SYSTEM [NOW24]) (RELEASE-TURN SYSTEM [NOW24])
```

Turn Holder: Manager
DU Stack:

```
DU-6  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G18980>
      :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047][ST-ACCEPT-0048])
      :PROCESSED ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047][ST-ACCEPT-0048])
DU-5  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G17094>
      :CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])
DU-4  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G13784>
      :CSA-LIST ([ST-SUGGEST-0032] [ST-CHECK-0033] [ST-ACCEPT-0034])
```

Figure A.13: Discourse Context after interpreting utterance 16

A.11 Interpreting Utterance 17

USER: how does that sound?

This utterance is interpreted as a request ([ST-REQUEST-0049]) to evaluate the preceding plan discussion. The content of the request is shown in (103).

(103)
```
(:LAMBDA ?A*T-EVAL
        (:AND (:ROLE ?A*T-EVAL :R-SPEAKER [SYS])
              (:ROLE ?A*T-EVAL :R-HEARER [HUM])
              (:CONTENT ?A*T-EVAL
                   (:LF-EXISTS ?RATE*T-ACCEPTABILITY-RATING ?DM*T-LIST NIL
                       (:THE ?P*T-PLAN ?DMPLAN*T-LIST
                             (:CURRENT-PLAN ?P*T-PLAN)
                             (:EVALUATION [SYS] ?P*T-PLAN ?RATE*T-ACCEPTABILITY-RATING))))))
```

Since [ST-REQUEST-0049] is a conversational level request, the domain plan reasoner is not consulted in verifying the act. In the current implementation all conversational level requests are deemed applicable, and so the final set of conversation acts is given in (104).

(104) :CSAS [ST-REQUEST-0049] :ARGS NIL
:GAS (:INIT MANAGER DU-7 ([ST-REQUEST-0049]) NIL)
:TTAS (KEEP-TURN MANAGER [NOW25]) (RELEASE-TURN MANAGER [NOW25])

The Resulting discourse context is shown in Figure A.14.

Turn Holder: System
DU Stack:
DU-7 :INITIATOR MANAGER :STATE 1 :CONTEXT #<context SB-G23720>
:CSA-LIST ([ST-REQUEST-0049]) :PROCESSED NIL
DU-6 :INITIATOR MANAGER :STATE F :CONTEXT #<context SB-G18980>
:CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047][ST-ACCEPT-0048])
DU-5 :INITIATOR MANAGER :STATE F :CONTEXT #<context SB-G17094>
:CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])

Figure A.14: Discourse Context after interpreting utterance 17

A.12 Producing and interpreting utterance 18-3

The system first decides to acknowledge DU-7, which adds the discourse obligation in (105). In order to meet the obligation, the dialogue manager calls the domain plan reasoner with an *elaborate-plan* call, to search for any problems or incomplete parts in the plan. The domain plan reasoner call returns that the plan is fine as it is, and so the dialogue manager decides to perform the requested action – an evaluation speech act. This is then generated as 18-3. The discourse state just before performing the utterance is shown in Figure A.15.

Discourse Obligations: (ADDRESS [ST-REQUEST-0049])
Turn Holder: System
Intended Speech Acts: (:ACK ([ST-REQUEST-0049]) GROUNDING)
(:EVAL (NO-PROBLEM [PLAN-4719]) (OBLIGATION (ADDRESS [ST-REQUEST-0049])))
DU Stack:
DU-7 :INITIATOR MANAGER :STATE 1 :CONTEXT #<context SB-G23720>
:CSA-LIST ([ST-REQUEST-0049]) :PROCESSED ([ST-REQUEST-0049])
DU-6 :INITIATOR MANAGER :STATE F :CONTEXT #<context SB-G18980>
:CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047][ST-ACCEPT-0048])
DU-5 :INITIATOR MANAGER :STATE F :CONTEXT #<context SB-G17094>
:CSA-LIST ([ST-INFORM-0037] [ST-SUGGEST-0040] [ST-SUPP-SUG-0041][ST-ACCEPT-0042])

Figure A.15: Discourse Context before producing utterance 18-3

(105) (ADDRESS [ST-REQUEST-0049])

SYSTEM: That's no problem.

Two core speech act hypotheses are formed for this utterance, shown in (106)a: an accept, with content shown in (106)b, and an evaluation, with content shown in (106)c.

```
(106) a. (:SURF-INTERP [CE2141] (:OR [ST-ACCEPT-0050] [ST-EVAL-0051]))

      b. (:LAMBDA ?SA*T-SPEECHACT
            (:AND (:ROLE ?SA*T-SPEECHACT :R-SPEAKER [HUM])
                  (:ROLE ?SA*T-SPEECHACT :R-HEARER [SYS]) (:OCCURS ?SA*T-SPEECHACT)
                  (:UNACCEPTED ?SA*T-SPEECHACT)))

      c. (:THE ?P*T-PLAN ?DMPLAN*T-LIST (:CURRENT-PLAN ?P*T-PLAN)
            (:EVALUATION [SYS] ?P*T-PLAN :NO-PROBLEM))
```

Evaluating [ST-ACCEPT-0050] shows this as accepting [ST-REQUEST-0049]. Evaluating [ST-EVAL-0051] means checking to see if there is a current plan which can be the object of an evaluation. This is found, so both interpretations are accepted.

This utterance produces two grounding acts: an acknowledgement of DU-7, as well as an initiation of DU-8, containing the evaluation. The conversation acts for this utterance are shown in (107), with the resulting discourse context in Figure A.16.

```
(107) :CSAS [ST-ACCEPT-0050] [ST-EVAL-0051] :ARGS NIL
      :GAS (:ACK SYSTEM DU-7 ([ST-ACCEPT-0050]) NIL)
           (:INIT SYSTEM DU-8 ([ST-EVAL-0051]) NIL)
      :TTAS (KEEP-TURN SYSTEM [NOW26]) (RELEASE-TURN SYSTEM [NOW26])
```

Turn Holder: Manager
DU Stack:

```
DU-8  :INITIATOR SYSTEM :STATE 1        :CONTEXT #<context SB-G6036>
      :CSA-LIST ([ST-EVAL-0051]) :PROCESSED NIL
DU-7  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G23720>
      :CSA-LIST ([ST-REQUEST-0049] [ST-ACCEPT-0050])
DU-6  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G18980>
      :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047][ST-ACCEPT-0048])
```

Figure A.16: Discourse Context after interpreting utterance 18-3

A.13 Interpreting Utterance 19

USER: good.

The only core speech act interpretation found for this utterance is an accept, with the same content as that shown in (29). This utterance is seen as accepting [ST-EVAL-0051]. The conversation acts for this utterance are shown in (108).

```
(108) :CSAS [ST-ACCEPT-0052] :ARGS NIL
      :GAS (:ACK MANAGER DU-8 ([ST-ACCEPT-0052]) NIL)
      :TTAS (TAKE-TURN NIL [NOW27]) (RELEASE-TURN NIL [NOW27])
```

Figure A.17 shows the resulting discourse context.

Now, finally, the initiative is given over to the system. Since nothing is more pressing, the system examines its discourse goals. The top level goal `have-plan-conversation` is expanded into the list shown in (109).

```
Turn Holder: System
DU Stack:
 DU-8 :INITIATOR SYSTEM :STATE F        :CONTEXT #<context SB-G6036>
       :CSA-LIST ([ST-EVAL-0051] [ST-ACCEPT-0052])
       :PROCESSED ([ST-EVAL-0051] [ST-ACCEPT-0052])
 DU-7  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G23720>
       :CSA-LIST ([ST-REQUEST-0049] [ST-ACCEPT-0050])
 DU-6  :INITIATOR MANAGER :STATE F       :CONTEXT #<context SB-G18980>
       :CSA-LIST ([ST-REQUEST-0044] [ST-REQUEST-0045] [ST-REQUEST-0047][ST-ACCEPT-0048])
Discourse Goals: have-plan-conversation
```

Figure A.17: Discourse Context after interpreting utterance 19

(109) `get-goal work-out-plan execute-plan`

`get-goal` was already accomplished after the first exchange when the user presented the goal for the conversation. `work-out-plan` was accomplished with the construction of a domain plan deemed successful. Since the system has access to its prior evaluation of the plan, it now moves on to the final goal `EXECUTE-PLAN`. This results in a call to the *domain plan executor* to execute `[PLAN-4719]` in the domain. The domain plan executor will transform the plan into a series of calls to the agents in the domain, and monitor for success. When it returns, the dialogue manager again deliberates, but this time there are no more goals, so it prints a closing message and terminates.

SYSTEM: The plan has been fully specified and executed.
Goodbye.